Content from the zip file `None`:

## File: word/CONTRATO PAGARE PLANTILLA.docx

CONTRATO PAGARE PLANTILLA

EMITIDO POR EMPRESA X

Contrato No. ${num\_contrato}

Entre los suscritos, **EMPRESA X,** sociedad domiciliada en CIUDAD, identificada con el RUC: **99999999999** con dirección xxxxxxx xxxxxxx xxxxxxxxx, empresa constituida en el PAIS, representada legalmente por **JEFFERSON JIMENEZ** identificado con DNI N° **123456789**. Quien en adelante se denominará **LA EMPRESA** y: **${nombre\_afiliado}**, identificado(a) con DNI: **${dni\_afiliado}** domiciliado en: **${dirección\_afiliado},** quien en adelante se denominará **AFILIADO**, celebrando el presente contrato de compra venta afiliaciones emitidas, el cual se regirá por las siguientes clausulas:

**CLAUSULA**: Los derechos surgidos de la adquisición de la afiliación, la calidad de afiliado y de beneficiarios tendrán una duración de **${anos\_afiliacion}**.

**CLAUSULA:** El precio de la acción de afiliación es de **${importe\_usd},** más la cuota administrativa de **${cuota\_adm\_usd},** valor que deberá ser cancelado al momento de la suscripción del presente contrato de compra venta de afiliaciones, el precio se anuncia en moneda extranjera y el afiliado acepta el pago en dicha moneda.

**CLAUSULA** El presente Contrato se entenderá perfeccionado con la firma de las partes. Conformes las partes con lo estipulado en el presente Contrato, lo aprueban y firman como aparece, en dos (02) ejemplares del mismo tenor, en la ciudad de CIUDAD, **${fecha\_contrato}**.

**ANEXO II:**

**PAGARE CONTRATO N° ${num\_contrato}**

Pagaré N° **${num\_pagare}**

Importe deudor **${adeuda\_usd}**

Vence el **${fecha\_pagare}**

Yo , **${nombre\_afiliado}** con DNI: **${dni\_afiliado}** reconozco que adeudo y pagaré incondicionalmente en la fecha de vencimiento consignado en el presente Pagaré, a la orden de **EMPRESA X**. en adelante denominada con nombre comercial **LA EMPRESA**, o a quien ésta se lo hubiera cedido, en su domicilio social o donde se presentare para su cobro; el importe de **${adeuda\_usd},** sin lugar a reclamo de clase alguna, para cuyo fiel y exacto cumplimiento, me obligo con todos mis bienes presente y futuros en la mejor forma de derecho. Al efecto, asumo la obligación en las siguientes condiciones.

Firma

**${nombre\_afiliado}**

**${dni\_afiliado}**

## File: word/contrato.sql

CREATE TABLE `contrato` (
  `num_contrato` bigint(30) NOT NULL,
  `nombre_afiliado` varchar(200) NOT NULL,
  `dni_afiliado` int(9) NOT NULL,
  `dirección_afiliado` text NOT NULL,
  `anos_afiliacion` int(3) NOT NULL,
  `importe_usd` decimal(30,2) NOT NULL,
  `cuota_adm_usd` decimal(30,2) NOT NULL,
  `fecha_contrato` date NOT NULL,
  `num_pagare` int(5) NOT NULL,
  `adeuda_usd` decimal(30,2) NOT NULL,
  `fecha_pagare` date NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4;

INSERT INTO `contrato` (`num_contrato`, `nombre_afiliado`, `dni_afiliado`, `dirección_afiliado`, `anos_afiliacion`, `importe_usd`, `cuota_adm_usd`, `fecha_contrato`, `num_pagare`, `adeuda_usd`, `fecha_pagare`) VALUES
(6503262, 'GILBERTO ANTONIO COLMENAREZ DAVILA', 654654654, 'Av. Micaela Cuellar # 0492 Piso 26', 6, '324.98', '12.63', '2022-12-01', 3654, '502.36', '2023-01-19'),
(6465412, 'MARIA JOSEFINA BUSTAMANTE GONZALEZ', 546513215, 'Callejón Aaron Torres, 5, Apto 1', 3, '1236.44', '36.99', '2022-11-16', 3254, '503.36', '2022-12-29');
ALTER TABLE `contrato`
  ADD PRIMARY KEY (`num_contrato`);
ALTER TABLE `contrato`
  MODIFY `num_contrato` bigint(30) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6503263;
COMMIT;

## File: word/index.php

Seleccione un contrato
GILBERTO
MARIA

## File: word/PHP_Word/composer.json

{
    "name": "user/word-php",
    "require": {
        "phpoffice/phpword": "^0.18.2"
    },
    "autoload": {
        "psr-4": {
            "User\\WordPhp\\": "src/"
        }
    },
    "authors": [
        {
            "name": "jefferson",
            "email": "jeff@gmail.com"
        }
    ]
}

## File: word/PHP_Word/composer.lock

{
    "_readme": [
        "This file locks the dependencies of your project to a known state",
        "Read more about it at https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies",
        "This file is @generated automatically"
    ],
    "content-hash": "49db745eab49fca88c791a562208a976",
    "packages": [
        {
            "name": "laminas/laminas-escaper",
            "version": "2.9.0",
            "source": {
                "type": "git",
                "url": "https://github.com/laminas/laminas-escaper.git",
                "reference": "891ad70986729e20ed2e86355fcf93c9dc238a5f"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/laminas/laminas-escaper/zipball/891ad70986729e20ed2e86355fcf93c9dc238a5f",
                "reference": "891ad70986729e20ed2e86355fcf93c9dc238a5f",
                "shasum": ""
            },
            "require": {
                "php": "^7.3 || ~8.0.0 || ~8.1.0"
            },
            "conflict": {
                "zendframework/zend-escaper": "*"
            },
            "require-dev": {
                "laminas/laminas-coding-standard": "~2.3.0",
                "phpunit/phpunit": "^9.3",
                "psalm/plugin-phpunit": "^0.12.2",
                "vimeo/psalm": "^3.16"
            },
            "suggest": {
                "ext-iconv": "*",
                "ext-mbstring": "*"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Laminas\\Escaper\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "description": "Securely and safely escape HTML, HTML attributes, JavaScript, CSS, and URLs",
            "homepage": "https://laminas.dev",
            "keywords": [
                "escaper",
                "laminas"
            ],
            "support": {
                "chat": "https://laminas.dev/chat",
                "docs": "https://docs.laminas.dev/laminas-escaper/",
                "forum": "https://discourse.laminas.dev",
                "issues": "https://github.com/laminas/laminas-escaper/issues",
                "rss": "https://github.com/laminas/laminas-escaper/releases.atom",
                "source": "https://github.com/laminas/laminas-escaper"
            },
            "funding": [
                {
                    "url": "https://funding.communitybridge.org/projects/laminas-project",
                    "type": "community_bridge"
                }
            ],
            "time": "2021-09-02T17:10:53+00:00"
        },
        {
            "name": "phpoffice/phpword",
            "version": "0.18.2",
            "source": {
                "type": "git",
                "url": "https://github.com/PHPOffice/PHPWord.git",
                "reference": "aca10785cf68dc95d7f6fac4fe854979fef3f8db"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/PHPOffice/PHPWord/zipball/aca10785cf68dc95d7f6fac4fe854979fef3f8db",
                "reference": "aca10785cf68dc95d7f6fac4fe854979fef3f8db",
                "shasum": ""
            },
            "require": {
                "ext-xml": "*",
                "laminas/laminas-escaper": "^2.2",
                "php": "^5.3.3 || ^7.0 || ^8.0"
            },
            "require-dev": {
                "dompdf/dompdf": "0.8.* || 1.0.*",
                "ext-gd": "*",
                "ext-zip": "*",
                "friendsofphp/php-cs-fixer": "^2.2",
                "mpdf/mpdf": "5.7.4 || 6.* || 7.* || 8.*",
                "php-coveralls/php-coveralls": "1.1.0 || ^2.0",
                "phploc/phploc": "2.* || 3.* || 4.* || 5.* || 6.* || 7.*",
                "phpmd/phpmd": "2.*",
                "phpunit/phpunit": "^4.8.36 || ^7.0",
                "squizlabs/php_codesniffer": "^2.9 || ^3.5",
                "tecnickcom/tcpdf": "6.*"
            },
            "suggest": {
                "dompdf/dompdf": "Allows writing PDF",
                "ext-gd2": "Allows adding images",
                "ext-xmlwriter": "Allows writing OOXML and ODF",
                "ext-xsl": "Allows applying XSL style sheet to headers, to main document part, and to footers of an OOXML template",
                "ext-zip": "Allows writing OOXML and ODF"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-develop": "0.19-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "PhpOffice\\PhpWord\\": "src/PhpWord"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "LGPL-3.0"
            ],
            "authors": [
                {
                    "name": "Mark Baker"
                },
                {
                    "name": "Gabriel Bull",
                    "email": "me@gabrielbull.com",
                    "homepage": "http://gabrielbull.com/"
                },
                {
                    "name": "Franck Lefevre",
                    "homepage": "https://rootslabs.net/blog/"
                },
                {
                    "name": "Ivan Lanin",
                    "homepage": "http://ivan.lanin.org"
                },
                {
                    "name": "Roman Syroeshko",
                    "homepage": "http://ru.linkedin.com/pub/roman-syroeshko/34/a53/994/"
                },
                {
                    "name": "Antoine de Troostembergh"
                }
            ],
            "description": "PHPWord - A pure PHP library for reading and writing word processing documents (OOXML, ODF, RTF, HTML, PDF)",
            "homepage": "http://phpoffice.github.io",
            "keywords": [
                "ISO IEC 29500",
                "OOXML",
                "Office Open XML",
                "OpenDocument",
                "OpenXML",
                "PhpOffice",
                "PhpWord",
                "Rich Text Format",
                "WordprocessingML",
                "doc",
                "docx",
                "html",
                "odf",
                "odt",
                "office",
                "pdf",
                "php",
                "reader",
                "rtf",
                "template",
                "template processor",
                "word",
                "writer"
            ],
            "support": {
                "issues": "https://github.com/PHPOffice/PHPWord/issues",
                "source": "https://github.com/PHPOffice/PHPWord/tree/0.18.2"
            },
            "time": "2021-06-04T20:58:45+00:00"
        }
    ],
    "packages-dev": [],
    "aliases": [],
    "minimum-stability": "stable",
    "stability-flags": [],
    "prefer-stable": false,
    "prefer-lowest": false,
    "platform": [],
    "platform-dev": [],
    "plugin-api-version": "2.1.0"
}

## File: word/PHP_Word/vendor/autoload.php

<?php

// autoload.php @generated by Composer

require_once __DIR__ . '/composer/autoload_real.php';

return ComposerAutoloaderInitd2d0822fa12307cbb1923b00116e942c::getLoader();

## File: word/PHP_Word/vendor/composer/autoload_classmap.php

<?php

// autoload_classmap.php @generated by Composer

$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);

return array(
    'Composer\\InstalledVersions' => $vendorDir . '/composer/InstalledVersions.php',
);

## File: word/PHP_Word/vendor/composer/autoload_namespaces.php

<?php

// autoload_namespaces.php @generated by Composer

$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);

return array(
);

## File: word/PHP_Word/vendor/composer/autoload_psr4.php

<?php

// autoload_psr4.php @generated by Composer

$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);

return array(
    'User\\WordPhp\\' => array($baseDir . '/src'),
    'PhpOffice\\PhpWord\\' => array($vendorDir . '/phpoffice/phpword/src/PhpWord'),
    'Laminas\\Escaper\\' => array($vendorDir . '/laminas/laminas-escaper/src'),
);

## File: word/PHP_Word/vendor/composer/autoload_real.php

<?php

// autoload_real.php @generated by Composer

class ComposerAutoloaderInitd2d0822fa12307cbb1923b00116e942c
{
    private static $loader;

    public static function loadClassLoader($class)
    {
        if ('Composer\Autoload\ClassLoader' === $class) {
            require __DIR__ . '/ClassLoader.php';
        }
    }

    /**
     * @return \Composer\Autoload\ClassLoader
     */
    public static function getLoader()
    {
        if (null !== self::$loader) {
            return self::$loader;
        }

        require __DIR__ . '/platform_check.php';

        spl_autoload_register(array('ComposerAutoloaderInitd2d0822fa12307cbb1923b00116e942c', 'loadClassLoader'), true, true);
        self::$loader = $loader = new \Composer\Autoload\ClassLoader(\dirname(\dirname(__FILE__)));
        spl_autoload_unregister(array('ComposerAutoloaderInitd2d0822fa12307cbb1923b00116e942c', 'loadClassLoader'));

        $useStaticLoader = PHP_VERSION_ID >= 50600 && !defined('HHVM_VERSION') && (!function_exists('zend_loader_file_encoded') || !zend_loader_file_encoded());
        if ($useStaticLoader) {
            require __DIR__ . '/autoload_static.php';

            call_user_func(\Composer\Autoload\ComposerStaticInitd2d0822fa12307cbb1923b00116e942c::getInitializer($loader));
        } else {
            $map = require __DIR__ . '/autoload_namespaces.php';
            foreach ($map as $namespace => $path) {
                $loader->set($namespace, $path);
            }

            $map = require __DIR__ . '/autoload_psr4.php';
            foreach ($map as $namespace => $path) {
                $loader->setPsr4($namespace, $path);
            }

            $classMap = require __DIR__ . '/autoload_classmap.php';
            if ($classMap) {
                $loader->addClassMap($classMap);
            }
        }

        $loader->register(true);

        return $loader;
    }
}

## File: word/PHP_Word/vendor/composer/autoload_static.php

<?php

// autoload_static.php @generated by Composer

namespace Composer\Autoload;

class ComposerStaticInitd2d0822fa12307cbb1923b00116e942c
{
    public static $prefixLengthsPsr4 = array (
        'U' =>
        array (
            'User\\WordPhp\\' => 13,
        ),
        'P' =>
        array (
            'PhpOffice\\PhpWord\\' => 18,
        ),
        'L' =>
        array (
            'Laminas\\Escaper\\' => 16,
        ),
    );

    public static $prefixDirsPsr4 = array (
        'User\\WordPhp\\' =>
        array (
            0 => __DIR__ . '/../..' . '/src',
        ),
        'PhpOffice\\PhpWord\\' =>
        array (
            0 => __DIR__ . '/..' . '/phpoffice/phpword/src/PhpWord',
        ),
        'Laminas\\Escaper\\' =>
        array (
            0 => __DIR__ . '/..' . '/laminas/laminas-escaper/src',
        ),
    );

    public static $classMap = array (
        'Composer\\InstalledVersions' => __DIR__ . '/..' . '/composer/InstalledVersions.php',
    );

    public static function getInitializer(ClassLoader $loader)
    {
        return \Closure::bind(function () use ($loader) {
            $loader->prefixLengthsPsr4 = ComposerStaticInitd2d0822fa12307cbb1923b00116e942c::$prefixLengthsPsr4;
            $loader->prefixDirsPsr4 = ComposerStaticInitd2d0822fa12307cbb1923b00116e942c::$prefixDirsPsr4;
            $loader->classMap = ComposerStaticInitd2d0822fa12307cbb1923b00116e942c::$classMap;

        }, null, ClassLoader::class);
    }
}

## File: word/PHP_Word/vendor/composer/ClassLoader.php

<?php

/*
 * This file is part of Composer.
 *
 * (c) Nils Adermann <naderman@naderman.de>
 *     Jordi Boggiano <j.boggiano@seld.be>
 *
 * For the full copyright and license information, please view the LICENSE
 * file that was distributed with this source code.
 */

namespace Composer\Autoload;

/**
 * ClassLoader implements a PSR-0, PSR-4 and classmap class loader.
 *
 *     $loader = new \Composer\Autoload\ClassLoader();
 *
 *     // register classes with namespaces
 *     $loader->add('Symfony\Component', __DIR__.'/component');
 *     $loader->add('Symfony',           __DIR__.'/framework');
 *
 *     // activate the autoloader
 *     $loader->register();
 *
 *     // to enable searching the include path (eg. for PEAR packages)
 *     $loader->setUseIncludePath(true);
 *
 * In this example, if you try to use a class in the Symfony\Component
 * namespace or one of its children (Symfony\Component\Console for instance),
 * the autoloader will first look for the class under the component/
 * directory, and it will then fallback to the framework/ directory if not
 * found before giving up.
 *
 * This class is loosely based on the Symfony UniversalClassLoader.
 *
 * @author Fabien Potencier <fabien@symfony.com>
 * @author Jordi Boggiano <j.boggiano@seld.be>
 * @see    https://www.php-fig.org/psr/psr-0/
 * @see    https://www.php-fig.org/psr/psr-4/
 */
class ClassLoader
{
    /** @var ?string */
    private $vendorDir;

    // PSR-4
    /**
     * @var array[]
     * @psalm-var array<string, array<string, int>>
     */
    private $prefixLengthsPsr4 = array();
    /**
     * @var array[]
     * @psalm-var array<string, array<int, string>>
     */
    private $prefixDirsPsr4 = array();
    /**
     * @var array[]
     * @psalm-var array<string, string>
     */
    private $fallbackDirsPsr4 = array();

    // PSR-0
    /**
     * @var array[]
     * @psalm-var array<string, array<string, string[]>>
     */
    private $prefixesPsr0 = array();
    /**
     * @var array[]
     * @psalm-var array<string, string>
     */
    private $fallbackDirsPsr0 = array();

    /** @var bool */
    private $useIncludePath = false;

    /**
     * @var string[]
     * @psalm-var array<string, string>
     */
    private $classMap = array();

    /** @var bool */
    private $classMapAuthoritative = false;

    /**
     * @var bool[]
     * @psalm-var array<string, bool>
     */
    private $missingClasses = array();

    /** @var ?string */
    private $apcuPrefix;

    /**
     * @var self[]
     */
    private static $registeredLoaders = array();

    /**
     * @param ?string $vendorDir
     */
    public function __construct($vendorDir = null)
    {
        $this->vendorDir = $vendorDir;
    }

    /**
     * @return string[]
     */
    public function getPrefixes()
    {
        if (!empty($this->prefixesPsr0)) {
            return call_user_func_array('array_merge', array_values($this->prefixesPsr0));
        }

        return array();
    }

    /**
     * @return array[]
     * @psalm-return array<string, array<int, string>>
     */
    public function getPrefixesPsr4()
    {
        return $this->prefixDirsPsr4;
    }

    /**
     * @return array[]
     * @psalm-return array<string, string>
     */
    public function getFallbackDirs()
    {
        return $this->fallbackDirsPsr0;
    }

    /**
     * @return array[]
     * @psalm-return array<string, string>
     */
    public function getFallbackDirsPsr4()
    {
        return $this->fallbackDirsPsr4;
    }

    /**
     * @return string[] Array of classname => path
     * @psalm-var array<string, string>
     */
    public function getClassMap()
    {
        return $this->classMap;
    }

    /**
     * @param string[] $classMap Class to filename map
     * @psalm-param array<string, string> $classMap
     *
     * @return void
     */
    public function addClassMap(array $classMap)
    {
        if ($this->classMap) {
            $this->classMap = array_merge($this->classMap, $classMap);
        } else {
            $this->classMap = $classMap;
        }
    }

    /**
     * Registers a set of PSR-0 directories for a given prefix, either
     * appending or prepending to the ones previously set for this prefix.
     *
     * @param string          $prefix  The prefix
     * @param string[]|string $paths   The PSR-0 root directories
     * @param bool            $prepend Whether to prepend the directories
     *
     * @return void
     */
    public function add($prefix, $paths, $prepend = false)
    {
        if (!$prefix) {
            if ($prepend) {
                $this->fallbackDirsPsr0 = array_merge(
                    (array) $paths,
                    $this->fallbackDirsPsr0
                );
            } else {
                $this->fallbackDirsPsr0 = array_merge(
                    $this->fallbackDirsPsr0,
                    (array) $paths
                );
            }

            return;
        }

        $first = $prefix[0];
        if (!isset($this->prefixesPsr0[$first][$prefix])) {
            $this->prefixesPsr0[$first][$prefix] = (array) $paths;

            return;
        }
        if ($prepend) {
            $this->prefixesPsr0[$first][$prefix] = array_merge(
                (array) $paths,
                $this->prefixesPsr0[$first][$prefix]
            );
        } else {
            $this->prefixesPsr0[$first][$prefix] = array_merge(
                $this->prefixesPsr0[$first][$prefix],
                (array) $paths
            );
        }
    }

    /**
     * Registers a set of PSR-4 directories for a given namespace, either
     * appending or prepending to the ones previously set for this namespace.
     *
     * @param string          $prefix  The prefix/namespace, with trailing '\\'
     * @param string[]|string $paths   The PSR-4 base directories
     * @param bool            $prepend Whether to prepend the directories
     *
     * @throws \InvalidArgumentException
     *
     * @return void
     */
    public function addPsr4($prefix, $paths, $prepend = false)
    {
        if (!$prefix) {
            // Register directories for the root namespace.
            if ($prepend) {
                $this->fallbackDirsPsr4 = array_merge(
                    (array) $paths,
                    $this->fallbackDirsPsr4
                );
            } else {
                $this->fallbackDirsPsr4 = array_merge(
                    $this->fallbackDirsPsr4,
                    (array) $paths
                );
            }
        } elseif (!isset($this->prefixDirsPsr4[$prefix])) {
            // Register directories for a new namespace.
            $length = strlen($prefix);
            if ('\\' !== $prefix[$length - 1]) {
                throw new \InvalidArgumentException("A non-empty PSR-4 prefix must end with a namespace separator.");
            }
            $this->prefixLengthsPsr4[$prefix[0]][$prefix] = $length;
            $this->prefixDirsPsr4[$prefix] = (array) $paths;
        } elseif ($prepend) {
            // Prepend directories for an already registered namespace.
            $this->prefixDirsPsr4[$prefix] = array_merge(
                (array) $paths,
                $this->prefixDirsPsr4[$prefix]
            );
        } else {
            // Append directories for an already registered namespace.
            $this->prefixDirsPsr4[$prefix] = array_merge(
                $this->prefixDirsPsr4[$prefix],
                (array) $paths
            );
        }
    }

    /**
     * Registers a set of PSR-0 directories for a given prefix,
     * replacing any others previously set for this prefix.
     *
     * @param string          $prefix The prefix
     * @param string[]|string $paths  The PSR-0 base directories
     *
     * @return void
     */
    public function set($prefix, $paths)
    {
        if (!$prefix) {
            $this->fallbackDirsPsr0 = (array) $paths;
        } else {
            $this->prefixesPsr0[$prefix[0]][$prefix] = (array) $paths;
        }
    }

    /**
     * Registers a set of PSR-4 directories for a given namespace,
     * replacing any others previously set for this namespace.
     *
     * @param string          $prefix The prefix/namespace, with trailing '\\'
     * @param string[]|string $paths  The PSR-4 base directories
     *
     * @throws \InvalidArgumentException
     *
     * @return void
     */
    public function setPsr4($prefix, $paths)
    {
        if (!$prefix) {
            $this->fallbackDirsPsr4 = (array) $paths;
        } else {
            $length = strlen($prefix);
            if ('\\' !== $prefix[$length - 1]) {
                throw new \InvalidArgumentException("A non-empty PSR-4 prefix must end with a namespace separator.");
            }
            $this->prefixLengthsPsr4[$prefix[0]][$prefix] = $length;
            $this->prefixDirsPsr4[$prefix] = (array) $paths;
        }
    }

    /**
     * Turns on searching the include path for class files.
     *
     * @param bool $useIncludePath
     *
     * @return void
     */
    public function setUseIncludePath($useIncludePath)
    {
        $this->useIncludePath = $useIncludePath;
    }

    /**
     * Can be used to check if the autoloader uses the include path to check
     * for classes.
     *
     * @return bool
     */
    public function getUseIncludePath()
    {
        return $this->useIncludePath;
    }

    /**
     * Turns off searching the prefix and fallback directories for classes
     * that have not been registered with the class map.
     *
     * @param bool $classMapAuthoritative
     *
     * @return void
     */
    public function setClassMapAuthoritative($classMapAuthoritative)
    {
        $this->classMapAuthoritative = $classMapAuthoritative;
    }

    /**
     * Should class lookup fail if not found in the current class map?
     *
     * @return bool
     */
    public function isClassMapAuthoritative()
    {
        return $this->classMapAuthoritative;
    }

    /**
     * APCu prefix to use to cache found/not-found classes, if the extension is enabled.
     *
     * @param string|null $apcuPrefix
     *
     * @return void
     */
    public function setApcuPrefix($apcuPrefix)
    {
        $this->apcuPrefix = function_exists('apcu_fetch') && filter_var(ini_get('apc.enabled'), FILTER_VALIDATE_BOOLEAN) ? $apcuPrefix : null;
    }

    /**
     * The APCu prefix in use, or null if APCu caching is not enabled.
     *
     * @return string|null
     */
    public function getApcuPrefix()
    {
        return $this->apcuPrefix;
    }

    /**
     * Registers this instance as an autoloader.
     *
     * @param bool $prepend Whether to prepend the autoloader or not
     *
     * @return void
     */
    public function register($prepend = false)
    {
        spl_autoload_register(array($this, 'loadClass'), true, $prepend);

        if (null === $this->vendorDir) {
            return;
        }

        if ($prepend) {
            self::$registeredLoaders = array($this->vendorDir => $this) + self::$registeredLoaders;
        } else {
            unset(self::$registeredLoaders[$this->vendorDir]);
            self::$registeredLoaders[$this->vendorDir] = $this;
        }
    }

    /**
     * Unregisters this instance as an autoloader.
     *
     * @return void
     */
    public function unregister()
    {
        spl_autoload_unregister(array($this, 'loadClass'));

        if (null !== $this->vendorDir) {
            unset(self::$registeredLoaders[$this->vendorDir]);
        }
    }

    /**
     * Loads the given class or interface.
     *
     * @param  string    $class The name of the class
     * @return true|null True if loaded, null otherwise
     */
    public function loadClass($class)
    {
        if ($file = $this->findFile($class)) {
            includeFile($file);

            return true;
        }

        return null;
    }

    /**
     * Finds the path to the file where the class is defined.
     *
     * @param string $class The name of the class
     *
     * @return string|false The path if found, false otherwise
     */
    public function findFile($class)
    {
        // class map lookup
        if (isset($this->classMap[$class])) {
            return $this->classMap[$class];
        }
        if ($this->classMapAuthoritative || isset($this->missingClasses[$class])) {
            return false;
        }
        if (null !== $this->apcuPrefix) {
            $file = apcu_fetch($this->apcuPrefix.$class, $hit);
            if ($hit) {
                return $file;
            }
        }

        $file = $this->findFileWithExtension($class, '.php');

        // Search for Hack files if we are running on HHVM
        if (false === $file && defined('HHVM_VERSION')) {
            $file = $this->findFileWithExtension($class, '.hh');
        }

        if (null !== $this->apcuPrefix) {
            apcu_add($this->apcuPrefix.$class, $file);
        }

        if (false === $file) {
            // Remember that this class does not exist.
            $this->missingClasses[$class] = true;
        }

        return $file;
    }

    /**
     * Returns the currently registered loaders indexed by their corresponding vendor directories.
     *
     * @return self[]
     */
    public static function getRegisteredLoaders()
    {
        return self::$registeredLoaders;
    }

    /**
     * @param  string       $class
     * @param  string       $ext
     * @return string|false
     */
    private function findFileWithExtension($class, $ext)
    {
        // PSR-4 lookup
        $logicalPathPsr4 = strtr($class, '\\', DIRECTORY_SEPARATOR) . $ext;

        $first = $class[0];
        if (isset($this->prefixLengthsPsr4[$first])) {
            $subPath = $class;
            while (false !== $lastPos = strrpos($subPath, '\\')) {
                $subPath = substr($subPath, 0, $lastPos);
                $search = $subPath . '\\';
                if (isset($this->prefixDirsPsr4[$search])) {
                    $pathEnd = DIRECTORY_SEPARATOR . substr($logicalPathPsr4, $lastPos + 1);
                    foreach ($this->prefixDirsPsr4[$search] as $dir) {
                        if (file_exists($file = $dir . $pathEnd)) {
                            return $file;
                        }
                    }
                }
            }
        }

        // PSR-4 fallback dirs
        foreach ($this->fallbackDirsPsr4 as $dir) {
            if (file_exists($file = $dir . DIRECTORY_SEPARATOR . $logicalPathPsr4)) {
                return $file;
            }
        }

        // PSR-0 lookup
        if (false !== $pos = strrpos($class, '\\')) {
            // namespaced class name
            $logicalPathPsr0 = substr($logicalPathPsr4, 0, $pos + 1)
                . strtr(substr($logicalPathPsr4, $pos + 1), '_', DIRECTORY_SEPARATOR);
        } else {
            // PEAR-like class name
            $logicalPathPsr0 = strtr($class, '_', DIRECTORY_SEPARATOR) . $ext;
        }

        if (isset($this->prefixesPsr0[$first])) {
            foreach ($this->prefixesPsr0[$first] as $prefix => $dirs) {
                if (0 === strpos($class, $prefix)) {
                    foreach ($dirs as $dir) {
                        if (file_exists($file = $dir . DIRECTORY_SEPARATOR . $logicalPathPsr0)) {
                            return $file;
                        }
                    }
                }
            }
        }

        // PSR-0 fallback dirs
        foreach ($this->fallbackDirsPsr0 as $dir) {
            if (file_exists($file = $dir . DIRECTORY_SEPARATOR . $logicalPathPsr0)) {
                return $file;
            }
        }

        // PSR-0 include paths.
        if ($this->useIncludePath && $file = stream_resolve_include_path($logicalPathPsr0)) {
            return $file;
        }

        return false;
    }
}

/**
 * Scope isolated include.
 *
 * Prevents access to $this/self from included files.
 *
 * @param  string $file
 * @return void
 * @private
 */
function includeFile($file)
{
    include $file;
}

## File: word/PHP_Word/vendor/composer/installed.json

{
    "packages": [
        {
            "name": "laminas/laminas-escaper",
            "version": "2.9.0",
            "version_normalized": "2.9.0.0",
            "source": {
                "type": "git",
                "url": "https://github.com/laminas/laminas-escaper.git",
                "reference": "891ad70986729e20ed2e86355fcf93c9dc238a5f"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/laminas/laminas-escaper/zipball/891ad70986729e20ed2e86355fcf93c9dc238a5f",
                "reference": "891ad70986729e20ed2e86355fcf93c9dc238a5f",
                "shasum": ""
            },
            "require": {
                "php": "^7.3 || ~8.0.0 || ~8.1.0"
            },
            "conflict": {
                "zendframework/zend-escaper": "*"
            },
            "require-dev": {
                "laminas/laminas-coding-standard": "~2.3.0",
                "phpunit/phpunit": "^9.3",
                "psalm/plugin-phpunit": "^0.12.2",
                "vimeo/psalm": "^3.16"
            },
            "suggest": {
                "ext-iconv": "*",
                "ext-mbstring": "*"
            },
            "time": "2021-09-02T17:10:53+00:00",
            "type": "library",
            "installation-source": "dist",
            "autoload": {
                "psr-4": {
                    "Laminas\\Escaper\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "description": "Securely and safely escape HTML, HTML attributes, JavaScript, CSS, and URLs",
            "homepage": "https://laminas.dev",
            "keywords": [
                "escaper",
                "laminas"
            ],
            "support": {
                "chat": "https://laminas.dev/chat",
                "docs": "https://docs.laminas.dev/laminas-escaper/",
                "forum": "https://discourse.laminas.dev",
                "issues": "https://github.com/laminas/laminas-escaper/issues",
                "rss": "https://github.com/laminas/laminas-escaper/releases.atom",
                "source": "https://github.com/laminas/laminas-escaper"
            },
            "funding": [
                {
                    "url": "https://funding.communitybridge.org/projects/laminas-project",
                    "type": "community_bridge"
                }
            ],
            "install-path": "../laminas/laminas-escaper"
        },
        {
            "name": "phpoffice/phpword",
            "version": "0.18.2",
            "version_normalized": "0.18.2.0",
            "source": {
                "type": "git",
                "url": "https://github.com/PHPOffice/PHPWord.git",
                "reference": "aca10785cf68dc95d7f6fac4fe854979fef3f8db"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/PHPOffice/PHPWord/zipball/aca10785cf68dc95d7f6fac4fe854979fef3f8db",
                "reference": "aca10785cf68dc95d7f6fac4fe854979fef3f8db",
                "shasum": ""
            },
            "require": {
                "ext-xml": "*",
                "laminas/laminas-escaper": "^2.2",
                "php": "^5.3.3 || ^7.0 || ^8.0"
            },
            "require-dev": {
                "dompdf/dompdf": "0.8.* || 1.0.*",
                "ext-gd": "*",
                "ext-zip": "*",
                "friendsofphp/php-cs-fixer": "^2.2",
                "mpdf/mpdf": "5.7.4 || 6.* || 7.* || 8.*",
                "php-coveralls/php-coveralls": "1.1.0 || ^2.0",
                "phploc/phploc": "2.* || 3.* || 4.* || 5.* || 6.* || 7.*",
                "phpmd/phpmd": "2.*",
                "phpunit/phpunit": "^4.8.36 || ^7.0",
                "squizlabs/php_codesniffer": "^2.9 || ^3.5",
                "tecnickcom/tcpdf": "6.*"
            },
            "suggest": {
                "dompdf/dompdf": "Allows writing PDF",
                "ext-gd2": "Allows adding images",
                "ext-xmlwriter": "Allows writing OOXML and ODF",
                "ext-xsl": "Allows applying XSL style sheet to headers, to main document part, and to footers of an OOXML template",
                "ext-zip": "Allows writing OOXML and ODF"
            },
            "time": "2021-06-04T20:58:45+00:00",
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-develop": "0.19-dev"
                }
            },
            "installation-source": "dist",
            "autoload": {
                "psr-4": {
                    "PhpOffice\\PhpWord\\": "src/PhpWord"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "LGPL-3.0"
            ],
            "authors": [
                {
                    "name": "Mark Baker"
                },
                {
                    "name": "Gabriel Bull",
                    "email": "me@gabrielbull.com",
                    "homepage": "http://gabrielbull.com/"
                },
                {
                    "name": "Franck Lefevre",
                    "homepage": "https://rootslabs.net/blog/"
                },
                {
                    "name": "Ivan Lanin",
                    "homepage": "http://ivan.lanin.org"
                },
                {
                    "name": "Roman Syroeshko",
                    "homepage": "http://ru.linkedin.com/pub/roman-syroeshko/34/a53/994/"
                },
                {
                    "name": "Antoine de Troostembergh"
                }
            ],
            "description": "PHPWord - A pure PHP library for reading and writing word processing documents (OOXML, ODF, RTF, HTML, PDF)",
            "homepage": "http://phpoffice.github.io",
            "keywords": [
                "ISO IEC 29500",
                "OOXML",
                "Office Open XML",
                "OpenDocument",
                "OpenXML",
                "PhpOffice",
                "PhpWord",
                "Rich Text Format",
                "WordprocessingML",
                "doc",
                "docx",
                "html",
                "odf",
                "odt",
                "office",
                "pdf",
                "php",
                "reader",
                "rtf",
                "template",
                "template processor",
                "word",
                "writer"
            ],
            "support": {
                "issues": "https://github.com/PHPOffice/PHPWord/issues",
                "source": "https://github.com/PHPOffice/PHPWord/tree/0.18.2"
            },
            "install-path": "../phpoffice/phpword"
        }
    ],
    "dev": true,
    "dev-package-names": []
}

## File: word/PHP_Word/vendor/composer/installed.php

<?php return array(
    'root' => array(
        'pretty_version' => '1.0.0+no-version-set',
        'version' => '1.0.0.0',
        'type' => 'library',
        'install_path' => __DIR__ . '/../../',
        'aliases' => array(),
        'reference' => NULL,
        'name' => 'user/word-php',
        'dev' => true,
    ),
    'versions' => array(
        'laminas/laminas-escaper' => array(
            'pretty_version' => '2.9.0',
            'version' => '2.9.0.0',
            'type' => 'library',
            'install_path' => __DIR__ . '/../laminas/laminas-escaper',
            'aliases' => array(),
            'reference' => '891ad70986729e20ed2e86355fcf93c9dc238a5f',
            'dev_requirement' => false,
        ),
        'phpoffice/phpword' => array(
            'pretty_version' => '0.18.2',
            'version' => '0.18.2.0',
            'type' => 'library',
            'install_path' => __DIR__ . '/../phpoffice/phpword',
            'aliases' => array(),
            'reference' => 'aca10785cf68dc95d7f6fac4fe854979fef3f8db',
            'dev_requirement' => false,
        ),
        'user/word-php' => array(
            'pretty_version' => '1.0.0+no-version-set',
            'version' => '1.0.0.0',
            'type' => 'library',
            'install_path' => __DIR__ . '/../../',
            'aliases' => array(),
            'reference' => NULL,
            'dev_requirement' => false,
        ),
    ),
);

## File: word/PHP_Word/vendor/composer/InstalledVersions.php

<?php

/*
 * This file is part of Composer.
 *
 * (c) Nils Adermann <naderman@naderman.de>
 *     Jordi Boggiano <j.boggiano@seld.be>
 *
 * For the full copyright and license information, please view the LICENSE
 * file that was distributed with this source code.
 */

namespace Composer;

use Composer\Autoload\ClassLoader;
use Composer\Semver\VersionParser;

/**
 * This class is copied in every Composer installed project and available to all
 *
 * See also https://getcomposer.org/doc/07-runtime.md#installed-versions
 *
 * To require its presence, you can require `composer-runtime-api ^2.0`
 */
class InstalledVersions
{
    /**
     * @var mixed[]|null
     * @psalm-var array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>}|array{}|null
     */
    private static $installed;

    /**
     * @var bool|null
     */
    private static $canGetVendors;

    /**
     * @var array[]
     * @psalm-var array<string, array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>}>
     */
    private static $installedByVendor = array();

    /**
     * Returns a list of all package names which are present, either by being installed, replaced or provided
     *
     * @return string[]
     * @psalm-return list<string>
     */
    public static function getInstalledPackages()
    {
        $packages = array();
        foreach (self::getInstalled() as $installed) {
            $packages[] = array_keys($installed['versions']);
        }

        if (1 === \count($packages)) {
            return $packages[0];
        }

        return array_keys(array_flip(\call_user_func_array('array_merge', $packages)));
    }

    /**
     * Returns a list of all package names with a specific type e.g. 'library'
     *
     * @param  string   $type
     * @return string[]
     * @psalm-return list<string>
     */
    public static function getInstalledPackagesByType($type)
    {
        $packagesByType = array();

        foreach (self::getInstalled() as $installed) {
            foreach ($installed['versions'] as $name => $package) {
                if (isset($package['type']) && $package['type'] === $type) {
                    $packagesByType[] = $name;
                }
            }
        }

        return $packagesByType;
    }

    /**
     * Checks whether the given package is installed
     *
     * This also returns true if the package name is provided or replaced by another package
     *
     * @param  string $packageName
     * @param  bool   $includeDevRequirements
     * @return bool
     */
    public static function isInstalled($packageName, $includeDevRequirements = true)
    {
        foreach (self::getInstalled() as $installed) {
            if (isset($installed['versions'][$packageName])) {
                return $includeDevRequirements || empty($installed['versions'][$packageName]['dev_requirement']);
            }
        }

        return false;
    }

    /**
     * Checks whether the given package satisfies a version constraint
     *
     * e.g. If you want to know whether version 2.3+ of package foo/bar is installed, you would call:
     *
     *   Composer\InstalledVersions::satisfies(new VersionParser, 'foo/bar', '^2.3')
     *
     * @param  VersionParser $parser      Install composer/semver to have access to this class and functionality
     * @param  string        $packageName
     * @param  string|null   $constraint  A version constraint to check for, if you pass one you have to make sure composer/semver is required by your package
     * @return bool
     */
    public static function satisfies(VersionParser $parser, $packageName, $constraint)
    {
        $constraint = $parser->parseConstraints($constraint);
        $provided = $parser->parseConstraints(self::getVersionRanges($packageName));

        return $provided->matches($constraint);
    }

    /**
     * Returns a version constraint representing all the range(s) which are installed for a given package
     *
     * It is easier to use this via isInstalled() with the $constraint argument if you need to check
     * whether a given version of a package is installed, and not just whether it exists
     *
     * @param  string $packageName
     * @return string Version constraint usable with composer/semver
     */
    public static function getVersionRanges($packageName)
    {
        foreach (self::getInstalled() as $installed) {
            if (!isset($installed['versions'][$packageName])) {
                continue;
            }

            $ranges = array();
            if (isset($installed['versions'][$packageName]['pretty_version'])) {
                $ranges[] = $installed['versions'][$packageName]['pretty_version'];
            }
            if (array_key_exists('aliases', $installed['versions'][$packageName])) {
                $ranges = array_merge($ranges, $installed['versions'][$packageName]['aliases']);
            }
            if (array_key_exists('replaced', $installed['versions'][$packageName])) {
                $ranges = array_merge($ranges, $installed['versions'][$packageName]['replaced']);
            }
            if (array_key_exists('provided', $installed['versions'][$packageName])) {
                $ranges = array_merge($ranges, $installed['versions'][$packageName]['provided']);
            }

            return implode(' || ', $ranges);
        }

        throw new \OutOfBoundsException('Package "' . $packageName . '" is not installed');
    }

    /**
     * @param  string      $packageName
     * @return string|null If the package is being replaced or provided but is not really installed, null will be returned as version, use satisfies or getVersionRanges if you need to know if a given version is present
     */
    public static function getVersion($packageName)
    {
        foreach (self::getInstalled() as $installed) {
            if (!isset($installed['versions'][$packageName])) {
                continue;
            }

            if (!isset($installed['versions'][$packageName]['version'])) {
                return null;
            }

            return $installed['versions'][$packageName]['version'];
        }

        throw new \OutOfBoundsException('Package "' . $packageName . '" is not installed');
    }

    /**
     * @param  string      $packageName
     * @return string|null If the package is being replaced or provided but is not really installed, null will be returned as version, use satisfies or getVersionRanges if you need to know if a given version is present
     */
    public static function getPrettyVersion($packageName)
    {
        foreach (self::getInstalled() as $installed) {
            if (!isset($installed['versions'][$packageName])) {
                continue;
            }

            if (!isset($installed['versions'][$packageName]['pretty_version'])) {
                return null;
            }

            return $installed['versions'][$packageName]['pretty_version'];
        }

        throw new \OutOfBoundsException('Package "' . $packageName . '" is not installed');
    }

    /**
     * @param  string      $packageName
     * @return string|null If the package is being replaced or provided but is not really installed, null will be returned as reference
     */
    public static function getReference($packageName)
    {
        foreach (self::getInstalled() as $installed) {
            if (!isset($installed['versions'][$packageName])) {
                continue;
            }

            if (!isset($installed['versions'][$packageName]['reference'])) {
                return null;
            }

            return $installed['versions'][$packageName]['reference'];
        }

        throw new \OutOfBoundsException('Package "' . $packageName . '" is not installed');
    }

    /**
     * @param  string      $packageName
     * @return string|null If the package is being replaced or provided but is not really installed, null will be returned as install path. Packages of type metapackages also have a null install path.
     */
    public static function getInstallPath($packageName)
    {
        foreach (self::getInstalled() as $installed) {
            if (!isset($installed['versions'][$packageName])) {
                continue;
            }

            return isset($installed['versions'][$packageName]['install_path']) ? $installed['versions'][$packageName]['install_path'] : null;
        }

        throw new \OutOfBoundsException('Package "' . $packageName . '" is not installed');
    }

    /**
     * @return array
     * @psalm-return array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}
     */
    public static function getRootPackage()
    {
        $installed = self::getInstalled();

        return $installed[0]['root'];
    }

    /**
     * Returns the raw installed.php data for custom implementations
     *
     * @deprecated Use getAllRawData() instead which returns all datasets for all autoloaders present in the process. getRawData only returns the first dataset loaded, which may not be what you expect.
     * @return array[]
     * @psalm-return array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>}
     */
    public static function getRawData()
    {
        @trigger_error('getRawData only returns the first dataset loaded, which may not be what you expect. Use getAllRawData() instead which returns all datasets for all autoloaders present in the process.', E_USER_DEPRECATED);

        if (null === self::$installed) {
            // only require the installed.php file if this file is loaded from its dumped location,
            // and not from its source location in the composer/composer package, see https://github.com/composer/composer/issues/9937
            if (substr(__DIR__, -8, 1) !== 'C') {
                self::$installed = include __DIR__ . '/installed.php';
            } else {
                self::$installed = array();
            }
        }

        return self::$installed;
    }

    /**
     * Returns the raw data of all installed.php which are currently loaded for custom implementations
     *
     * @return array[]
     * @psalm-return list<array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>}>
     */
    public static function getAllRawData()
    {
        return self::getInstalled();
    }

    /**
     * Lets you reload the static array from another file
     *
     * This is only useful for complex integrations in which a project needs to use
     * this class but then also needs to execute another project's autoloader in process,
     * and wants to ensure both projects have access to their version of installed.php.
     *
     * A typical case would be PHPUnit, where it would need to make sure it reads all
     * the data it needs from this class, then call reload() with
     * `require $CWD/vendor/composer/installed.php` (or similar) as input to make sure
     * the project in which it runs can then also use this class safely, without
     * interference between PHPUnit's dependencies and the project's dependencies.
     *
     * @param  array[] $data A vendor/composer/installed.php data set
     * @return void
     *
     * @psalm-param array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>} $data
     */
    public static function reload($data)
    {
        self::$installed = $data;
        self::$installedByVendor = array();
    }

    /**
     * @return array[]
     * @psalm-return list<array{root: array{name: string, version: string, reference: string, pretty_version: string, aliases: string[], dev: bool, install_path: string, type: string}, versions: array<string, array{dev_requirement: bool, pretty_version?: string, version?: string, aliases?: string[], reference?: string, replaced?: string[], provided?: string[], install_path?: string, type?: string}>}>
     */
    private static function getInstalled()
    {
        if (null === self::$canGetVendors) {
            self::$canGetVendors = method_exists('Composer\Autoload\ClassLoader', 'getRegisteredLoaders');
        }

        $installed = array();

        if (self::$canGetVendors) {
            foreach (ClassLoader::getRegisteredLoaders() as $vendorDir => $loader) {
                if (isset(self::$installedByVendor[$vendorDir])) {
                    $installed[] = self::$installedByVendor[$vendorDir];
                } elseif (is_file($vendorDir.'/composer/installed.php')) {
                    $installed[] = self::$installedByVendor[$vendorDir] = require $vendorDir.'/composer/installed.php';
                    if (null === self::$installed && strtr($vendorDir.'/composer', '\\', '/') === strtr(__DIR__, '\\', '/')) {
                        self::$installed = $installed[count($installed) - 1];
                    }
                }
            }
        }

        if (null === self::$installed) {
            // only require the installed.php file if this file is loaded from its dumped location,
            // and not from its source location in the composer/composer package, see https://github.com/composer/composer/issues/9937
            if (substr(__DIR__, -8, 1) !== 'C') {
                self::$installed = require __DIR__ . '/installed.php';
            } else {
                self::$installed = array();
            }
        }
        $installed[] = self::$installed;

        return $installed;
    }
}

## File: word/PHP_Word/vendor/composer/LICENSE

Copyright (c) Nils Adermann, Jordi Boggiano

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is furnished
to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

## File: word/PHP_Word/vendor/composer/platform_check.php

<?php

// platform_check.php @generated by Composer

$issues = array();

if (!(PHP_VERSION_ID >= 70300)) {
    $issues[] = 'Your Composer dependencies require a PHP version ">= 7.3.0". You are running ' . PHP_VERSION . '.';
}

if ($issues) {
    if (!headers_sent()) {
        header('HTTP/1.1 500 Internal Server Error');
    }
    if (!ini_get('display_errors')) {
        if (PHP_SAPI === 'cli' || PHP_SAPI === 'phpdbg') {
            fwrite(STDERR, 'Composer detected issues in your platform:' . PHP_EOL.PHP_EOL . implode(PHP_EOL, $issues) . PHP_EOL.PHP_EOL);
        } elseif (!headers_sent()) {
            echo 'Composer detected issues in your platform:' . PHP_EOL.PHP_EOL . str_replace('You are running '.PHP_VERSION.'.', '', implode(PHP_EOL, $issues)) . PHP_EOL.PHP_EOL;
        }
    }
    trigger_error(
        'Composer detected issues in your platform: ' . implode(' ', $issues),
        E_USER_ERROR
    );
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/.laminas-ci.json

{
    "ignore_php_platform_requirements": {
        "8.1": true
    }
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/composer.json

{
    "name": "laminas/laminas-escaper",
    "description": "Securely and safely escape HTML, HTML attributes, JavaScript, CSS, and URLs",
    "license": "BSD-3-Clause",
    "keywords": [
        "laminas",
        "escaper"
    ],
    "homepage": "https://laminas.dev",
    "support": {
        "docs": "https://docs.laminas.dev/laminas-escaper/",
        "issues": "https://github.com/laminas/laminas-escaper/issues",
        "source": "https://github.com/laminas/laminas-escaper",
        "rss": "https://github.com/laminas/laminas-escaper/releases.atom",
        "chat": "https://laminas.dev/chat",
        "forum": "https://discourse.laminas.dev"
    },
    "config": {
        "sort-packages": true
    },
    "extra": {
    },
    "require": {
        "php": "^7.3 || ~8.0.0 || ~8.1.0"
    },
    "suggest": {
        "ext-iconv": "*",
        "ext-mbstring": "*"
    },
    "require-dev": {
        "laminas/laminas-coding-standard": "~2.3.0",
        "phpunit/phpunit": "^9.3",
        "psalm/plugin-phpunit": "^0.12.2",
        "vimeo/psalm": "^3.16"
    },
    "autoload": {
        "psr-4": {
            "Laminas\\Escaper\\": "src/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "LaminasTest\\Escaper\\": "test/"
        }
    },
    "scripts": {
        "check": [
            "@cs-check",
            "@test"
        ],
        "cs-check": "phpcs",
        "cs-fix": "phpcbf",
        "static-analysis": "psalm --shepherd --stats",
        "test": "phpunit --colors=always",
        "test-coverage": "phpunit --colors=always --coverage-clover clover.xml"
    },
    "conflict": {
        "zendframework/zend-escaper": "*"
    }
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/composer.lock

{
    "_readme": [
        "This file locks the dependencies of your project to a known state",
        "Read more about it at https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies",
        "This file is @generated automatically"
    ],
    "content-hash": "5c54f272d6c88f5ee83f3aaa6a9ed107",
    "packages": [],
    "packages-dev": [
        {
            "name": "amphp/amp",
            "version": "v2.6.0",
            "source": {
                "type": "git",
                "url": "https://github.com/amphp/amp.git",
                "reference": "caa95edeb1ca1bf7532e9118ede4a3c3126408cc"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/amphp/amp/zipball/caa95edeb1ca1bf7532e9118ede4a3c3126408cc",
                "reference": "caa95edeb1ca1bf7532e9118ede4a3c3126408cc",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "require-dev": {
                "amphp/php-cs-fixer-config": "dev-master",
                "amphp/phpunit-util": "^1",
                "ext-json": "*",
                "jetbrains/phpstorm-stubs": "^2019.3",
                "phpunit/phpunit": "^7 | ^8 | ^9",
                "psalm/phar": "^3.11@dev",
                "react/promise": "^2"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Amp\\": "lib"
                },
                "files": [
                    "lib/functions.php",
                    "lib/Internal/functions.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Daniel Lowrey",
                    "email": "rdlowrey@php.net"
                },
                {
                    "name": "Aaron Piotrowski",
                    "email": "aaron@trowski.com"
                },
                {
                    "name": "Bob Weinand",
                    "email": "bobwei9@hotmail.com"
                },
                {
                    "name": "Niklas Keller",
                    "email": "me@kelunik.com"
                }
            ],
            "description": "A non-blocking concurrency framework for PHP applications.",
            "homepage": "http://amphp.org/amp",
            "keywords": [
                "async",
                "asynchronous",
                "awaitable",
                "concurrency",
                "event",
                "event-loop",
                "future",
                "non-blocking",
                "promise"
            ],
            "support": {
                "irc": "irc://irc.freenode.org/amphp",
                "issues": "https://github.com/amphp/amp/issues",
                "source": "https://github.com/amphp/amp/tree/v2.6.0"
            },
            "funding": [
                {
                    "url": "https://github.com/amphp",
                    "type": "github"
                }
            ],
            "time": "2021-07-16T20:06:06+00:00"
        },
        {
            "name": "amphp/byte-stream",
            "version": "v1.8.1",
            "source": {
                "type": "git",
                "url": "https://github.com/amphp/byte-stream.git",
                "reference": "acbd8002b3536485c997c4e019206b3f10ca15bd"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/amphp/byte-stream/zipball/acbd8002b3536485c997c4e019206b3f10ca15bd",
                "reference": "acbd8002b3536485c997c4e019206b3f10ca15bd",
                "shasum": ""
            },
            "require": {
                "amphp/amp": "^2",
                "php": ">=7.1"
            },
            "require-dev": {
                "amphp/php-cs-fixer-config": "dev-master",
                "amphp/phpunit-util": "^1.4",
                "friendsofphp/php-cs-fixer": "^2.3",
                "jetbrains/phpstorm-stubs": "^2019.3",
                "phpunit/phpunit": "^6 || ^7 || ^8",
                "psalm/phar": "^3.11.4"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Amp\\ByteStream\\": "lib"
                },
                "files": [
                    "lib/functions.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Aaron Piotrowski",
                    "email": "aaron@trowski.com"
                },
                {
                    "name": "Niklas Keller",
                    "email": "me@kelunik.com"
                }
            ],
            "description": "A stream abstraction to make working with non-blocking I/O simple.",
            "homepage": "http://amphp.org/byte-stream",
            "keywords": [
                "amp",
                "amphp",
                "async",
                "io",
                "non-blocking",
                "stream"
            ],
            "support": {
                "irc": "irc://irc.freenode.org/amphp",
                "issues": "https://github.com/amphp/byte-stream/issues",
                "source": "https://github.com/amphp/byte-stream/tree/v1.8.1"
            },
            "funding": [
                {
                    "url": "https://github.com/amphp",
                    "type": "github"
                }
            ],
            "time": "2021-03-30T17:13:30+00:00"
        },
        {
            "name": "composer/package-versions-deprecated",
            "version": "1.11.99.3",
            "source": {
                "type": "git",
                "url": "https://github.com/composer/package-versions-deprecated.git",
                "reference": "fff576ac850c045158a250e7e27666e146e78d18"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/composer/package-versions-deprecated/zipball/fff576ac850c045158a250e7e27666e146e78d18",
                "reference": "fff576ac850c045158a250e7e27666e146e78d18",
                "shasum": ""
            },
            "require": {
                "composer-plugin-api": "^1.1.0 || ^2.0",
                "php": "^7 || ^8"
            },
            "replace": {
                "ocramius/package-versions": "1.11.99"
            },
            "require-dev": {
                "composer/composer": "^1.9.3 || ^2.0@dev",
                "ext-zip": "^1.13",
                "phpunit/phpunit": "^6.5 || ^7"
            },
            "type": "composer-plugin",
            "extra": {
                "class": "PackageVersions\\Installer",
                "branch-alias": {
                    "dev-master": "1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "PackageVersions\\": "src/PackageVersions"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Marco Pivetta",
                    "email": "ocramius@gmail.com"
                },
                {
                    "name": "Jordi Boggiano",
                    "email": "j.boggiano@seld.be"
                }
            ],
            "description": "Composer plugin that provides efficient querying for installed package versions (no runtime IO)",
            "support": {
                "issues": "https://github.com/composer/package-versions-deprecated/issues",
                "source": "https://github.com/composer/package-versions-deprecated/tree/1.11.99.3"
            },
            "funding": [
                {
                    "url": "https://packagist.com",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/composer",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/composer/composer",
                    "type": "tidelift"
                }
            ],
            "time": "2021-08-17T13:49:14+00:00"
        },
        {
            "name": "composer/semver",
            "version": "3.2.5",
            "source": {
                "type": "git",
                "url": "https://github.com/composer/semver.git",
                "reference": "31f3ea725711245195f62e54ffa402d8ef2fdba9"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/composer/semver/zipball/31f3ea725711245195f62e54ffa402d8ef2fdba9",
                "reference": "31f3ea725711245195f62e54ffa402d8ef2fdba9",
                "shasum": ""
            },
            "require": {
                "php": "^5.3.2 || ^7.0 || ^8.0"
            },
            "require-dev": {
                "phpstan/phpstan": "^0.12.54",
                "symfony/phpunit-bridge": "^4.2 || ^5"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "3.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Composer\\Semver\\": "src"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nils Adermann",
                    "email": "naderman@naderman.de",
                    "homepage": "http://www.naderman.de"
                },
                {
                    "name": "Jordi Boggiano",
                    "email": "j.boggiano@seld.be",
                    "homepage": "http://seld.be"
                },
                {
                    "name": "Rob Bast",
                    "email": "rob.bast@gmail.com",
                    "homepage": "http://robbast.nl"
                }
            ],
            "description": "Semver library that offers utilities, version constraint parsing and validation.",
            "keywords": [
                "semantic",
                "semver",
                "validation",
                "versioning"
            ],
            "support": {
                "irc": "irc://irc.freenode.org/composer",
                "issues": "https://github.com/composer/semver/issues",
                "source": "https://github.com/composer/semver/tree/3.2.5"
            },
            "funding": [
                {
                    "url": "https://packagist.com",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/composer",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/composer/composer",
                    "type": "tidelift"
                }
            ],
            "time": "2021-05-24T12:41:47+00:00"
        },
        {
            "name": "composer/xdebug-handler",
            "version": "1.4.6",
            "source": {
                "type": "git",
                "url": "https://github.com/composer/xdebug-handler.git",
                "reference": "f27e06cd9675801df441b3656569b328e04aa37c"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/composer/xdebug-handler/zipball/f27e06cd9675801df441b3656569b328e04aa37c",
                "reference": "f27e06cd9675801df441b3656569b328e04aa37c",
                "shasum": ""
            },
            "require": {
                "php": "^5.3.2 || ^7.0 || ^8.0",
                "psr/log": "^1.0"
            },
            "require-dev": {
                "phpstan/phpstan": "^0.12.55",
                "symfony/phpunit-bridge": "^4.2 || ^5"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Composer\\XdebugHandler\\": "src"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "John Stevenson",
                    "email": "john-stevenson@blueyonder.co.uk"
                }
            ],
            "description": "Restarts a process without Xdebug.",
            "keywords": [
                "Xdebug",
                "performance"
            ],
            "support": {
                "irc": "irc://irc.freenode.org/composer",
                "issues": "https://github.com/composer/xdebug-handler/issues",
                "source": "https://github.com/composer/xdebug-handler/tree/1.4.6"
            },
            "funding": [
                {
                    "url": "https://packagist.com",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/composer",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/composer/composer",
                    "type": "tidelift"
                }
            ],
            "time": "2021-03-25T17:01:18+00:00"
        },
        {
            "name": "dealerdirect/phpcodesniffer-composer-installer",
            "version": "v0.7.1",
            "source": {
                "type": "git",
                "url": "https://github.com/Dealerdirect/phpcodesniffer-composer-installer.git",
                "reference": "fe390591e0241955f22eb9ba327d137e501c771c"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/Dealerdirect/phpcodesniffer-composer-installer/zipball/fe390591e0241955f22eb9ba327d137e501c771c",
                "reference": "fe390591e0241955f22eb9ba327d137e501c771c",
                "shasum": ""
            },
            "require": {
                "composer-plugin-api": "^1.0 || ^2.0",
                "php": ">=5.3",
                "squizlabs/php_codesniffer": "^2.0 || ^3.0 || ^4.0"
            },
            "require-dev": {
                "composer/composer": "*",
                "phpcompatibility/php-compatibility": "^9.0",
                "sensiolabs/security-checker": "^4.1.0"
            },
            "type": "composer-plugin",
            "extra": {
                "class": "Dealerdirect\\Composer\\Plugin\\Installers\\PHPCodeSniffer\\Plugin"
            },
            "autoload": {
                "psr-4": {
                    "Dealerdirect\\Composer\\Plugin\\Installers\\PHPCodeSniffer\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Franck Nijhof",
                    "email": "franck.nijhof@dealerdirect.com",
                    "homepage": "http://www.frenck.nl",
                    "role": "Developer / IT Manager"
                }
            ],
            "description": "PHP_CodeSniffer Standards Composer Installer Plugin",
            "homepage": "http://www.dealerdirect.com",
            "keywords": [
                "PHPCodeSniffer",
                "PHP_CodeSniffer",
                "code quality",
                "codesniffer",
                "composer",
                "installer",
                "phpcs",
                "plugin",
                "qa",
                "quality",
                "standard",
                "standards",
                "style guide",
                "stylecheck",
                "tests"
            ],
            "support": {
                "issues": "https://github.com/dealerdirect/phpcodesniffer-composer-installer/issues",
                "source": "https://github.com/dealerdirect/phpcodesniffer-composer-installer"
            },
            "time": "2020-12-07T18:04:37+00:00"
        },
        {
            "name": "dnoegel/php-xdg-base-dir",
            "version": "v0.1.1",
            "source": {
                "type": "git",
                "url": "https://github.com/dnoegel/php-xdg-base-dir.git",
                "reference": "8f8a6e48c5ecb0f991c2fdcf5f154a47d85f9ffd"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/dnoegel/php-xdg-base-dir/zipball/8f8a6e48c5ecb0f991c2fdcf5f154a47d85f9ffd",
                "reference": "8f8a6e48c5ecb0f991c2fdcf5f154a47d85f9ffd",
                "shasum": ""
            },
            "require": {
                "php": ">=5.3.2"
            },
            "require-dev": {
                "phpunit/phpunit": "~7.0|~6.0|~5.0|~4.8.35"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "XdgBaseDir\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "description": "implementation of xdg base directory specification for php",
            "support": {
                "issues": "https://github.com/dnoegel/php-xdg-base-dir/issues",
                "source": "https://github.com/dnoegel/php-xdg-base-dir/tree/v0.1.1"
            },
            "time": "2019-12-04T15:06:13+00:00"
        },
        {
            "name": "doctrine/instantiator",
            "version": "1.4.0",
            "source": {
                "type": "git",
                "url": "https://github.com/doctrine/instantiator.git",
                "reference": "d56bf6102915de5702778fe20f2de3b2fe570b5b"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/doctrine/instantiator/zipball/d56bf6102915de5702778fe20f2de3b2fe570b5b",
                "reference": "d56bf6102915de5702778fe20f2de3b2fe570b5b",
                "shasum": ""
            },
            "require": {
                "php": "^7.1 || ^8.0"
            },
            "require-dev": {
                "doctrine/coding-standard": "^8.0",
                "ext-pdo": "*",
                "ext-phar": "*",
                "phpbench/phpbench": "^0.13 || 1.0.0-alpha2",
                "phpstan/phpstan": "^0.12",
                "phpstan/phpstan-phpunit": "^0.12",
                "phpunit/phpunit": "^7.0 || ^8.0 || ^9.0"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Doctrine\\Instantiator\\": "src/Doctrine/Instantiator/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Marco Pivetta",
                    "email": "ocramius@gmail.com",
                    "homepage": "https://ocramius.github.io/"
                }
            ],
            "description": "A small, lightweight utility to instantiate objects in PHP without invoking their constructors",
            "homepage": "https://www.doctrine-project.org/projects/instantiator.html",
            "keywords": [
                "constructor",
                "instantiate"
            ],
            "support": {
                "issues": "https://github.com/doctrine/instantiator/issues",
                "source": "https://github.com/doctrine/instantiator/tree/1.4.0"
            },
            "funding": [
                {
                    "url": "https://www.doctrine-project.org/sponsorship.html",
                    "type": "custom"
                },
                {
                    "url": "https://www.patreon.com/phpdoctrine",
                    "type": "patreon"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/doctrine%2Finstantiator",
                    "type": "tidelift"
                }
            ],
            "time": "2020-11-10T18:47:58+00:00"
        },
        {
            "name": "felixfbecker/advanced-json-rpc",
            "version": "v3.2.1",
            "source": {
                "type": "git",
                "url": "https://github.com/felixfbecker/php-advanced-json-rpc.git",
                "reference": "b5f37dbff9a8ad360ca341f3240dc1c168b45447"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/felixfbecker/php-advanced-json-rpc/zipball/b5f37dbff9a8ad360ca341f3240dc1c168b45447",
                "reference": "b5f37dbff9a8ad360ca341f3240dc1c168b45447",
                "shasum": ""
            },
            "require": {
                "netresearch/jsonmapper": "^1.0 || ^2.0 || ^3.0 || ^4.0",
                "php": "^7.1 || ^8.0",
                "phpdocumentor/reflection-docblock": "^4.3.4 || ^5.0.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^7.0 || ^8.0"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "AdvancedJsonRpc\\": "lib/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "ISC"
            ],
            "authors": [
                {
                    "name": "Felix Becker",
                    "email": "felix.b@outlook.com"
                }
            ],
            "description": "A more advanced JSONRPC implementation",
            "support": {
                "issues": "https://github.com/felixfbecker/php-advanced-json-rpc/issues",
                "source": "https://github.com/felixfbecker/php-advanced-json-rpc/tree/v3.2.1"
            },
            "time": "2021-06-11T22:34:44+00:00"
        },
        {
            "name": "felixfbecker/language-server-protocol",
            "version": "1.5.1",
            "source": {
                "type": "git",
                "url": "https://github.com/felixfbecker/php-language-server-protocol.git",
                "reference": "9d846d1f5cf101deee7a61c8ba7caa0a975cd730"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/felixfbecker/php-language-server-protocol/zipball/9d846d1f5cf101deee7a61c8ba7caa0a975cd730",
                "reference": "9d846d1f5cf101deee7a61c8ba7caa0a975cd730",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "require-dev": {
                "phpstan/phpstan": "*",
                "squizlabs/php_codesniffer": "^3.1",
                "vimeo/psalm": "^4.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "LanguageServerProtocol\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "ISC"
            ],
            "authors": [
                {
                    "name": "Felix Becker",
                    "email": "felix.b@outlook.com"
                }
            ],
            "description": "PHP classes for the Language Server Protocol",
            "keywords": [
                "language",
                "microsoft",
                "php",
                "server"
            ],
            "support": {
                "issues": "https://github.com/felixfbecker/php-language-server-protocol/issues",
                "source": "https://github.com/felixfbecker/php-language-server-protocol/tree/1.5.1"
            },
            "time": "2021-02-22T14:02:09+00:00"
        },
        {
            "name": "laminas/laminas-coding-standard",
            "version": "2.3.0",
            "source": {
                "type": "git",
                "url": "https://github.com/laminas/laminas-coding-standard.git",
                "reference": "bcf6e07fe4690240be7beb6d884d0b0fafa6a251"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/laminas/laminas-coding-standard/zipball/bcf6e07fe4690240be7beb6d884d0b0fafa6a251",
                "reference": "bcf6e07fe4690240be7beb6d884d0b0fafa6a251",
                "shasum": ""
            },
            "require": {
                "dealerdirect/phpcodesniffer-composer-installer": "^0.7",
                "php": "^7.3 || ^8.0",
                "slevomat/coding-standard": "^7.0",
                "squizlabs/php_codesniffer": "^3.6",
                "webimpress/coding-standard": "^1.2"
            },
            "type": "phpcodesniffer-standard",
            "autoload": {
                "psr-4": {
                    "LaminasCodingStandard\\": "src/LaminasCodingStandard/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "description": "Laminas Coding Standard",
            "homepage": "https://laminas.dev",
            "keywords": [
                "Coding Standard",
                "laminas"
            ],
            "support": {
                "chat": "https://laminas.dev/chat",
                "docs": "https://docs.laminas.dev/laminas-coding-standard/",
                "forum": "https://discourse.laminas.dev",
                "issues": "https://github.com/laminas/laminas-coding-standard/issues",
                "rss": "https://github.com/laminas/laminas-coding-standard/releases.atom",
                "source": "https://github.com/laminas/laminas-coding-standard"
            },
            "funding": [
                {
                    "url": "https://funding.communitybridge.org/projects/laminas-project",
                    "type": "community_bridge"
                }
            ],
            "time": "2021-05-29T15:53:59+00:00"
        },
        {
            "name": "myclabs/deep-copy",
            "version": "1.10.2",
            "source": {
                "type": "git",
                "url": "https://github.com/myclabs/DeepCopy.git",
                "reference": "776f831124e9c62e1a2c601ecc52e776d8bb7220"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/myclabs/DeepCopy/zipball/776f831124e9c62e1a2c601ecc52e776d8bb7220",
                "reference": "776f831124e9c62e1a2c601ecc52e776d8bb7220",
                "shasum": ""
            },
            "require": {
                "php": "^7.1 || ^8.0"
            },
            "replace": {
                "myclabs/deep-copy": "self.version"
            },
            "require-dev": {
                "doctrine/collections": "^1.0",
                "doctrine/common": "^2.6",
                "phpunit/phpunit": "^7.1"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "DeepCopy\\": "src/DeepCopy/"
                },
                "files": [
                    "src/DeepCopy/deep_copy.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "description": "Create deep copies (clones) of your objects",
            "keywords": [
                "clone",
                "copy",
                "duplicate",
                "object",
                "object graph"
            ],
            "support": {
                "issues": "https://github.com/myclabs/DeepCopy/issues",
                "source": "https://github.com/myclabs/DeepCopy/tree/1.10.2"
            },
            "funding": [
                {
                    "url": "https://tidelift.com/funding/github/packagist/myclabs/deep-copy",
                    "type": "tidelift"
                }
            ],
            "time": "2020-11-13T09:40:50+00:00"
        },
        {
            "name": "netresearch/jsonmapper",
            "version": "v3.1.1",
            "source": {
                "type": "git",
                "url": "https://github.com/cweiske/jsonmapper.git",
                "reference": "ba09f0e456d4f00cef84e012da5715625594407c"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/cweiske/jsonmapper/zipball/ba09f0e456d4f00cef84e012da5715625594407c",
                "reference": "ba09f0e456d4f00cef84e012da5715625594407c",
                "shasum": ""
            },
            "require": {
                "ext-json": "*",
                "ext-pcre": "*",
                "ext-reflection": "*",
                "ext-spl": "*",
                "php": ">=5.6"
            },
            "require-dev": {
                "phpunit/phpunit": "~4.8.35 || ~5.7 || ~6.4 || ~7.0",
                "squizlabs/php_codesniffer": "~3.5"
            },
            "type": "library",
            "autoload": {
                "psr-0": {
                    "JsonMapper": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "OSL-3.0"
            ],
            "authors": [
                {
                    "name": "Christian Weiske",
                    "email": "cweiske@cweiske.de",
                    "homepage": "http://github.com/cweiske/jsonmapper/",
                    "role": "Developer"
                }
            ],
            "description": "Map nested JSON structures onto PHP classes",
            "support": {
                "email": "cweiske@cweiske.de",
                "issues": "https://github.com/cweiske/jsonmapper/issues",
                "source": "https://github.com/cweiske/jsonmapper/tree/v3.1.1"
            },
            "time": "2020-11-02T19:19:54+00:00"
        },
        {
            "name": "nikic/php-parser",
            "version": "v4.12.0",
            "source": {
                "type": "git",
                "url": "https://github.com/nikic/PHP-Parser.git",
                "reference": "6608f01670c3cc5079e18c1dab1104e002579143"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/nikic/PHP-Parser/zipball/6608f01670c3cc5079e18c1dab1104e002579143",
                "reference": "6608f01670c3cc5079e18c1dab1104e002579143",
                "shasum": ""
            },
            "require": {
                "ext-tokenizer": "*",
                "php": ">=7.0"
            },
            "require-dev": {
                "ircmaxell/php-yacc": "^0.0.7",
                "phpunit/phpunit": "^6.5 || ^7.0 || ^8.0 || ^9.0"
            },
            "bin": [
                "bin/php-parse"
            ],
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.9-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "PhpParser\\": "lib/PhpParser"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Nikita Popov"
                }
            ],
            "description": "A PHP parser written in PHP",
            "keywords": [
                "parser",
                "php"
            ],
            "support": {
                "issues": "https://github.com/nikic/PHP-Parser/issues",
                "source": "https://github.com/nikic/PHP-Parser/tree/v4.12.0"
            },
            "time": "2021-07-21T10:44:31+00:00"
        },
        {
            "name": "openlss/lib-array2xml",
            "version": "1.0.0",
            "source": {
                "type": "git",
                "url": "https://github.com/nullivex/lib-array2xml.git",
                "reference": "a91f18a8dfc69ffabe5f9b068bc39bb202c81d90"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/nullivex/lib-array2xml/zipball/a91f18a8dfc69ffabe5f9b068bc39bb202c81d90",
                "reference": "a91f18a8dfc69ffabe5f9b068bc39bb202c81d90",
                "shasum": ""
            },
            "require": {
                "php": ">=5.3.2"
            },
            "type": "library",
            "autoload": {
                "psr-0": {
                    "LSS": ""
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "Apache-2.0"
            ],
            "authors": [
                {
                    "name": "Bryan Tong",
                    "email": "bryan@nullivex.com",
                    "homepage": "https://www.nullivex.com"
                },
                {
                    "name": "Tony Butler",
                    "email": "spudz76@gmail.com",
                    "homepage": "https://www.nullivex.com"
                }
            ],
            "description": "Array2XML conversion library credit to lalit.org",
            "homepage": "https://www.nullivex.com",
            "keywords": [
                "array",
                "array conversion",
                "xml",
                "xml conversion"
            ],
            "support": {
                "issues": "https://github.com/nullivex/lib-array2xml/issues",
                "source": "https://github.com/nullivex/lib-array2xml/tree/master"
            },
            "time": "2019-03-29T20:06:56+00:00"
        },
        {
            "name": "phar-io/manifest",
            "version": "2.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/phar-io/manifest.git",
                "reference": "97803eca37d319dfa7826cc2437fc020857acb53"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phar-io/manifest/zipball/97803eca37d319dfa7826cc2437fc020857acb53",
                "reference": "97803eca37d319dfa7826cc2437fc020857acb53",
                "shasum": ""
            },
            "require": {
                "ext-dom": "*",
                "ext-phar": "*",
                "ext-xmlwriter": "*",
                "phar-io/version": "^3.0.1",
                "php": "^7.2 || ^8.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.0.x-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Arne Blankerts",
                    "email": "arne@blankerts.de",
                    "role": "Developer"
                },
                {
                    "name": "Sebastian Heuer",
                    "email": "sebastian@phpeople.de",
                    "role": "Developer"
                },
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "Developer"
                }
            ],
            "description": "Component for reading phar.io manifest information from a PHP Archive (PHAR)",
            "support": {
                "issues": "https://github.com/phar-io/manifest/issues",
                "source": "https://github.com/phar-io/manifest/tree/2.0.3"
            },
            "time": "2021-07-20T11:28:43+00:00"
        },
        {
            "name": "phar-io/version",
            "version": "3.1.0",
            "source": {
                "type": "git",
                "url": "https://github.com/phar-io/version.git",
                "reference": "bae7c545bef187884426f042434e561ab1ddb182"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phar-io/version/zipball/bae7c545bef187884426f042434e561ab1ddb182",
                "reference": "bae7c545bef187884426f042434e561ab1ddb182",
                "shasum": ""
            },
            "require": {
                "php": "^7.2 || ^8.0"
            },
            "type": "library",
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Arne Blankerts",
                    "email": "arne@blankerts.de",
                    "role": "Developer"
                },
                {
                    "name": "Sebastian Heuer",
                    "email": "sebastian@phpeople.de",
                    "role": "Developer"
                },
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "Developer"
                }
            ],
            "description": "Library for handling version information and constraints",
            "support": {
                "issues": "https://github.com/phar-io/version/issues",
                "source": "https://github.com/phar-io/version/tree/3.1.0"
            },
            "time": "2021-02-23T14:00:09+00:00"
        },
        {
            "name": "phpdocumentor/reflection-common",
            "version": "2.2.0",
            "source": {
                "type": "git",
                "url": "https://github.com/phpDocumentor/ReflectionCommon.git",
                "reference": "1d01c49d4ed62f25aa84a747ad35d5a16924662b"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phpDocumentor/ReflectionCommon/zipball/1d01c49d4ed62f25aa84a747ad35d5a16924662b",
                "reference": "1d01c49d4ed62f25aa84a747ad35d5a16924662b",
                "shasum": ""
            },
            "require": {
                "php": "^7.2 || ^8.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-2.x": "2.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "phpDocumentor\\Reflection\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Jaap van Otterdijk",
                    "email": "opensource@ijaap.nl"
                }
            ],
            "description": "Common reflection classes used by phpdocumentor to reflect the code structure",
            "homepage": "http://www.phpdoc.org",
            "keywords": [
                "FQSEN",
                "phpDocumentor",
                "phpdoc",
                "reflection",
                "static analysis"
            ],
            "support": {
                "issues": "https://github.com/phpDocumentor/ReflectionCommon/issues",
                "source": "https://github.com/phpDocumentor/ReflectionCommon/tree/2.x"
            },
            "time": "2020-06-27T09:03:43+00:00"
        },
        {
            "name": "phpdocumentor/reflection-docblock",
            "version": "5.2.2",
            "source": {
                "type": "git",
                "url": "https://github.com/phpDocumentor/ReflectionDocBlock.git",
                "reference": "069a785b2141f5bcf49f3e353548dc1cce6df556"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phpDocumentor/ReflectionDocBlock/zipball/069a785b2141f5bcf49f3e353548dc1cce6df556",
                "reference": "069a785b2141f5bcf49f3e353548dc1cce6df556",
                "shasum": ""
            },
            "require": {
                "ext-filter": "*",
                "php": "^7.2 || ^8.0",
                "phpdocumentor/reflection-common": "^2.2",
                "phpdocumentor/type-resolver": "^1.3",
                "webmozart/assert": "^1.9.1"
            },
            "require-dev": {
                "mockery/mockery": "~1.3.2"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "5.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "phpDocumentor\\Reflection\\": "src"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Mike van Riel",
                    "email": "me@mikevanriel.com"
                },
                {
                    "name": "Jaap van Otterdijk",
                    "email": "account@ijaap.nl"
                }
            ],
            "description": "With this component, a library can provide support for annotations via DocBlocks or otherwise retrieve information that is embedded in a DocBlock.",
            "support": {
                "issues": "https://github.com/phpDocumentor/ReflectionDocBlock/issues",
                "source": "https://github.com/phpDocumentor/ReflectionDocBlock/tree/master"
            },
            "time": "2020-09-03T19:13:55+00:00"
        },
        {
            "name": "phpdocumentor/type-resolver",
            "version": "1.4.0",
            "source": {
                "type": "git",
                "url": "https://github.com/phpDocumentor/TypeResolver.git",
                "reference": "6a467b8989322d92aa1c8bf2bebcc6e5c2ba55c0"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phpDocumentor/TypeResolver/zipball/6a467b8989322d92aa1c8bf2bebcc6e5c2ba55c0",
                "reference": "6a467b8989322d92aa1c8bf2bebcc6e5c2ba55c0",
                "shasum": ""
            },
            "require": {
                "php": "^7.2 || ^8.0",
                "phpdocumentor/reflection-common": "^2.0"
            },
            "require-dev": {
                "ext-tokenizer": "*"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-1.x": "1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "phpDocumentor\\Reflection\\": "src"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Mike van Riel",
                    "email": "me@mikevanriel.com"
                }
            ],
            "description": "A PSR-5 based resolver of Class names, Types and Structural Element Names",
            "support": {
                "issues": "https://github.com/phpDocumentor/TypeResolver/issues",
                "source": "https://github.com/phpDocumentor/TypeResolver/tree/1.4.0"
            },
            "time": "2020-09-17T18:55:26+00:00"
        },
        {
            "name": "phpspec/prophecy",
            "version": "1.13.0",
            "source": {
                "type": "git",
                "url": "https://github.com/phpspec/prophecy.git",
                "reference": "be1996ed8adc35c3fd795488a653f4b518be70ea"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phpspec/prophecy/zipball/be1996ed8adc35c3fd795488a653f4b518be70ea",
                "reference": "be1996ed8adc35c3fd795488a653f4b518be70ea",
                "shasum": ""
            },
            "require": {
                "doctrine/instantiator": "^1.2",
                "php": "^7.2 || ~8.0, <8.1",
                "phpdocumentor/reflection-docblock": "^5.2",
                "sebastian/comparator": "^3.0 || ^4.0",
                "sebastian/recursion-context": "^3.0 || ^4.0"
            },
            "require-dev": {
                "phpspec/phpspec": "^6.0",
                "phpunit/phpunit": "^8.0 || ^9.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.11.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Prophecy\\": "src/Prophecy"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Konstantin Kudryashov",
                    "email": "ever.zet@gmail.com",
                    "homepage": "http://everzet.com"
                },
                {
                    "name": "Marcello Duarte",
                    "email": "marcello.duarte@gmail.com"
                }
            ],
            "description": "Highly opinionated mocking framework for PHP 5.3+",
            "homepage": "https://github.com/phpspec/prophecy",
            "keywords": [
                "Double",
                "Dummy",
                "fake",
                "mock",
                "spy",
                "stub"
            ],
            "support": {
                "issues": "https://github.com/phpspec/prophecy/issues",
                "source": "https://github.com/phpspec/prophecy/tree/1.13.0"
            },
            "time": "2021-03-17T13:42:18+00:00"
        },
        {
            "name": "phpstan/phpdoc-parser",
            "version": "0.5.5",
            "source": {
                "type": "git",
                "url": "https://github.com/phpstan/phpdoc-parser.git",
                "reference": "ea0b17460ec38e20d7eb64e7ec49b5d44af5d28c"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/phpstan/phpdoc-parser/zipball/ea0b17460ec38e20d7eb64e7ec49b5d44af5d28c",
                "reference": "ea0b17460ec38e20d7eb64e7ec49b5d44af5d28c",
                "shasum": ""
            },
            "require": {
                "php": "^7.1 || ^8.0"
            },
            "require-dev": {
                "php-parallel-lint/php-parallel-lint": "^1.2",
                "phpstan/extension-installer": "^1.0",
                "phpstan/phpstan": "^0.12.87",
                "phpstan/phpstan-strict-rules": "^0.12.5",
                "phpunit/phpunit": "^9.5",
                "symfony/process": "^5.2"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "0.5-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "PHPStan\\PhpDocParser\\": [
                        "src/"
                    ]
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "description": "PHPDoc parser with support for nullable, intersection and generic types",
            "support": {
                "issues": "https://github.com/phpstan/phpdoc-parser/issues",
                "source": "https://github.com/phpstan/phpdoc-parser/tree/0.5.5"
            },
            "time": "2021-06-11T13:24:46+00:00"
        },
        {
            "name": "phpunit/php-code-coverage",
            "version": "9.2.6",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/php-code-coverage.git",
                "reference": "f6293e1b30a2354e8428e004689671b83871edde"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/php-code-coverage/zipball/f6293e1b30a2354e8428e004689671b83871edde",
                "reference": "f6293e1b30a2354e8428e004689671b83871edde",
                "shasum": ""
            },
            "require": {
                "ext-dom": "*",
                "ext-libxml": "*",
                "ext-xmlwriter": "*",
                "nikic/php-parser": "^4.10.2",
                "php": ">=7.3",
                "phpunit/php-file-iterator": "^3.0.3",
                "phpunit/php-text-template": "^2.0.2",
                "sebastian/code-unit-reverse-lookup": "^2.0.2",
                "sebastian/complexity": "^2.0",
                "sebastian/environment": "^5.1.2",
                "sebastian/lines-of-code": "^1.0.3",
                "sebastian/version": "^3.0.1",
                "theseer/tokenizer": "^1.2.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "suggest": {
                "ext-pcov": "*",
                "ext-xdebug": "*"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "9.2-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Library that provides collection, processing, and rendering functionality for PHP code coverage information.",
            "homepage": "https://github.com/sebastianbergmann/php-code-coverage",
            "keywords": [
                "coverage",
                "testing",
                "xunit"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/php-code-coverage/issues",
                "source": "https://github.com/sebastianbergmann/php-code-coverage/tree/9.2.6"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2021-03-28T07:26:59+00:00"
        },
        {
            "name": "phpunit/php-file-iterator",
            "version": "3.0.5",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/php-file-iterator.git",
                "reference": "aa4be8575f26070b100fccb67faabb28f21f66f8"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/php-file-iterator/zipball/aa4be8575f26070b100fccb67faabb28f21f66f8",
                "reference": "aa4be8575f26070b100fccb67faabb28f21f66f8",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "FilterIterator implementation that filters files based on a list of suffixes.",
            "homepage": "https://github.com/sebastianbergmann/php-file-iterator/",
            "keywords": [
                "filesystem",
                "iterator"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/php-file-iterator/issues",
                "source": "https://github.com/sebastianbergmann/php-file-iterator/tree/3.0.5"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T05:57:25+00:00"
        },
        {
            "name": "phpunit/php-invoker",
            "version": "3.1.1",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/php-invoker.git",
                "reference": "5a10147d0aaf65b58940a0b72f71c9ac0423cc67"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/php-invoker/zipball/5a10147d0aaf65b58940a0b72f71c9ac0423cc67",
                "reference": "5a10147d0aaf65b58940a0b72f71c9ac0423cc67",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "ext-pcntl": "*",
                "phpunit/phpunit": "^9.3"
            },
            "suggest": {
                "ext-pcntl": "*"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.1-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Invoke callables with a timeout",
            "homepage": "https://github.com/sebastianbergmann/php-invoker/",
            "keywords": [
                "process"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/php-invoker/issues",
                "source": "https://github.com/sebastianbergmann/php-invoker/tree/3.1.1"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T05:58:55+00:00"
        },
        {
            "name": "phpunit/php-text-template",
            "version": "2.0.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/php-text-template.git",
                "reference": "5da5f67fc95621df9ff4c4e5a84d6a8a2acf7c28"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/php-text-template/zipball/5da5f67fc95621df9ff4c4e5a84d6a8a2acf7c28",
                "reference": "5da5f67fc95621df9ff4c4e5a84d6a8a2acf7c28",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Simple template engine.",
            "homepage": "https://github.com/sebastianbergmann/php-text-template/",
            "keywords": [
                "template"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/php-text-template/issues",
                "source": "https://github.com/sebastianbergmann/php-text-template/tree/2.0.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T05:33:50+00:00"
        },
        {
            "name": "phpunit/php-timer",
            "version": "5.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/php-timer.git",
                "reference": "5a63ce20ed1b5bf577850e2c4e87f4aa902afbd2"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/php-timer/zipball/5a63ce20ed1b5bf577850e2c4e87f4aa902afbd2",
                "reference": "5a63ce20ed1b5bf577850e2c4e87f4aa902afbd2",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "5.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Utility class for timing",
            "homepage": "https://github.com/sebastianbergmann/php-timer/",
            "keywords": [
                "timer"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/php-timer/issues",
                "source": "https://github.com/sebastianbergmann/php-timer/tree/5.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:16:10+00:00"
        },
        {
            "name": "phpunit/phpunit",
            "version": "9.5.9",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/phpunit.git",
                "reference": "ea8c2dfb1065eb35a79b3681eee6e6fb0a6f273b"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/phpunit/zipball/ea8c2dfb1065eb35a79b3681eee6e6fb0a6f273b",
                "reference": "ea8c2dfb1065eb35a79b3681eee6e6fb0a6f273b",
                "shasum": ""
            },
            "require": {
                "doctrine/instantiator": "^1.3.1",
                "ext-dom": "*",
                "ext-json": "*",
                "ext-libxml": "*",
                "ext-mbstring": "*",
                "ext-xml": "*",
                "ext-xmlwriter": "*",
                "myclabs/deep-copy": "^1.10.1",
                "phar-io/manifest": "^2.0.3",
                "phar-io/version": "^3.0.2",
                "php": ">=7.3",
                "phpspec/prophecy": "^1.12.1",
                "phpunit/php-code-coverage": "^9.2.3",
                "phpunit/php-file-iterator": "^3.0.5",
                "phpunit/php-invoker": "^3.1.1",
                "phpunit/php-text-template": "^2.0.3",
                "phpunit/php-timer": "^5.0.2",
                "sebastian/cli-parser": "^1.0.1",
                "sebastian/code-unit": "^1.0.6",
                "sebastian/comparator": "^4.0.5",
                "sebastian/diff": "^4.0.3",
                "sebastian/environment": "^5.1.3",
                "sebastian/exporter": "^4.0.3",
                "sebastian/global-state": "^5.0.1",
                "sebastian/object-enumerator": "^4.0.3",
                "sebastian/resource-operations": "^3.0.3",
                "sebastian/type": "^2.3.4",
                "sebastian/version": "^3.0.2"
            },
            "require-dev": {
                "ext-pdo": "*",
                "phpspec/prophecy-phpunit": "^2.0.1"
            },
            "suggest": {
                "ext-soap": "*",
                "ext-xdebug": "*"
            },
            "bin": [
                "phpunit"
            ],
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "9.5-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ],
                "files": [
                    "src/Framework/Assert/Functions.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "The PHP Unit Testing framework.",
            "homepage": "https://phpunit.de/",
            "keywords": [
                "phpunit",
                "testing",
                "xunit"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/phpunit/issues",
                "source": "https://github.com/sebastianbergmann/phpunit/tree/9.5.9"
            },
            "funding": [
                {
                    "url": "https://phpunit.de/donate.html",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2021-08-31T06:47:40+00:00"
        },
        {
            "name": "psalm/plugin-phpunit",
            "version": "0.12.2",
            "source": {
                "type": "git",
                "url": "https://github.com/psalm/psalm-plugin-phpunit.git",
                "reference": "85ee5a080a5281e63085d933b30a06b1b1680758"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/psalm/psalm-plugin-phpunit/zipball/85ee5a080a5281e63085d933b30a06b1b1680758",
                "reference": "85ee5a080a5281e63085d933b30a06b1b1680758",
                "shasum": ""
            },
            "require": {
                "composer/package-versions-deprecated": "^1.10",
                "composer/semver": "^1.4 || ^2.0 || ^3.0",
                "ext-simplexml": "*",
                "php": "^7.1.3 || ^8.0",
                "phpunit/phpunit": "^7.5 || ^8.0 || ^9.0",
                "vimeo/psalm": "^3.6.2 || dev-master || dev-4.x"
            },
            "require-dev": {
                "codeception/codeception": "^4.0.3",
                "squizlabs/php_codesniffer": "^3.3.1",
                "weirdan/codeception-psalm-module": "^0.7.1",
                "weirdan/prophecy-shim": "^1.0 || ^2.0"
            },
            "type": "psalm-plugin",
            "extra": {
                "psalm": {
                    "pluginClass": "Psalm\\PhpUnitPlugin\\Plugin"
                }
            },
            "autoload": {
                "psr-4": {
                    "Psalm\\PhpUnitPlugin\\": "src"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Matt Brown",
                    "email": "github@muglug.com"
                }
            ],
            "description": "Psalm plugin for PHPUnit",
            "support": {
                "issues": "https://github.com/psalm/psalm-plugin-phpunit/issues",
                "source": "https://github.com/psalm/psalm-plugin-phpunit/tree/0.12.2"
            },
            "time": "2020-09-28T17:25:39+00:00"
        },
        {
            "name": "psr/container",
            "version": "1.1.1",
            "source": {
                "type": "git",
                "url": "https://github.com/php-fig/container.git",
                "reference": "8622567409010282b7aeebe4bb841fe98b58dcaf"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/php-fig/container/zipball/8622567409010282b7aeebe4bb841fe98b58dcaf",
                "reference": "8622567409010282b7aeebe4bb841fe98b58dcaf",
                "shasum": ""
            },
            "require": {
                "php": ">=7.2.0"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Psr\\Container\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "PHP-FIG",
                    "homepage": "https://www.php-fig.org/"
                }
            ],
            "description": "Common Container Interface (PHP FIG PSR-11)",
            "homepage": "https://github.com/php-fig/container",
            "keywords": [
                "PSR-11",
                "container",
                "container-interface",
                "container-interop",
                "psr"
            ],
            "support": {
                "issues": "https://github.com/php-fig/container/issues",
                "source": "https://github.com/php-fig/container/tree/1.1.1"
            },
            "time": "2021-03-05T17:36:06+00:00"
        },
        {
            "name": "psr/log",
            "version": "1.1.4",
            "source": {
                "type": "git",
                "url": "https://github.com/php-fig/log.git",
                "reference": "d49695b909c3b7628b6289db5479a1c204601f11"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/php-fig/log/zipball/d49695b909c3b7628b6289db5479a1c204601f11",
                "reference": "d49695b909c3b7628b6289db5479a1c204601f11",
                "shasum": ""
            },
            "require": {
                "php": ">=5.3.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Psr\\Log\\": "Psr/Log/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "PHP-FIG",
                    "homepage": "https://www.php-fig.org/"
                }
            ],
            "description": "Common interface for logging libraries",
            "homepage": "https://github.com/php-fig/log",
            "keywords": [
                "log",
                "psr",
                "psr-3"
            ],
            "support": {
                "source": "https://github.com/php-fig/log/tree/1.1.4"
            },
            "time": "2021-05-03T11:20:27+00:00"
        },
        {
            "name": "sebastian/cli-parser",
            "version": "1.0.1",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/cli-parser.git",
                "reference": "442e7c7e687e42adc03470c7b668bc4b2402c0b2"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/cli-parser/zipball/442e7c7e687e42adc03470c7b668bc4b2402c0b2",
                "reference": "442e7c7e687e42adc03470c7b668bc4b2402c0b2",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Library for parsing CLI options",
            "homepage": "https://github.com/sebastianbergmann/cli-parser",
            "support": {
                "issues": "https://github.com/sebastianbergmann/cli-parser/issues",
                "source": "https://github.com/sebastianbergmann/cli-parser/tree/1.0.1"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T06:08:49+00:00"
        },
        {
            "name": "sebastian/code-unit",
            "version": "1.0.8",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/code-unit.git",
                "reference": "1fc9f64c0927627ef78ba436c9b17d967e68e120"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/code-unit/zipball/1fc9f64c0927627ef78ba436c9b17d967e68e120",
                "reference": "1fc9f64c0927627ef78ba436c9b17d967e68e120",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Collection of value objects that represent the PHP code units",
            "homepage": "https://github.com/sebastianbergmann/code-unit",
            "support": {
                "issues": "https://github.com/sebastianbergmann/code-unit/issues",
                "source": "https://github.com/sebastianbergmann/code-unit/tree/1.0.8"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:08:54+00:00"
        },
        {
            "name": "sebastian/code-unit-reverse-lookup",
            "version": "2.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/code-unit-reverse-lookup.git",
                "reference": "ac91f01ccec49fb77bdc6fd1e548bc70f7faa3e5"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/code-unit-reverse-lookup/zipball/ac91f01ccec49fb77bdc6fd1e548bc70f7faa3e5",
                "reference": "ac91f01ccec49fb77bdc6fd1e548bc70f7faa3e5",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Looks up which function or method a line of code belongs to",
            "homepage": "https://github.com/sebastianbergmann/code-unit-reverse-lookup/",
            "support": {
                "issues": "https://github.com/sebastianbergmann/code-unit-reverse-lookup/issues",
                "source": "https://github.com/sebastianbergmann/code-unit-reverse-lookup/tree/2.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T05:30:19+00:00"
        },
        {
            "name": "sebastian/comparator",
            "version": "4.0.6",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/comparator.git",
                "reference": "55f4261989e546dc112258c7a75935a81a7ce382"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/comparator/zipball/55f4261989e546dc112258c7a75935a81a7ce382",
                "reference": "55f4261989e546dc112258c7a75935a81a7ce382",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3",
                "sebastian/diff": "^4.0",
                "sebastian/exporter": "^4.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                },
                {
                    "name": "Jeff Welch",
                    "email": "whatthejeff@gmail.com"
                },
                {
                    "name": "Volker Dusch",
                    "email": "github@wallbash.com"
                },
                {
                    "name": "Bernhard Schussek",
                    "email": "bschussek@2bepublished.at"
                }
            ],
            "description": "Provides the functionality to compare PHP values for equality",
            "homepage": "https://github.com/sebastianbergmann/comparator",
            "keywords": [
                "comparator",
                "compare",
                "equality"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/comparator/issues",
                "source": "https://github.com/sebastianbergmann/comparator/tree/4.0.6"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T15:49:45+00:00"
        },
        {
            "name": "sebastian/complexity",
            "version": "2.0.2",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/complexity.git",
                "reference": "739b35e53379900cc9ac327b2147867b8b6efd88"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/complexity/zipball/739b35e53379900cc9ac327b2147867b8b6efd88",
                "reference": "739b35e53379900cc9ac327b2147867b8b6efd88",
                "shasum": ""
            },
            "require": {
                "nikic/php-parser": "^4.7",
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Library for calculating the complexity of PHP code units",
            "homepage": "https://github.com/sebastianbergmann/complexity",
            "support": {
                "issues": "https://github.com/sebastianbergmann/complexity/issues",
                "source": "https://github.com/sebastianbergmann/complexity/tree/2.0.2"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T15:52:27+00:00"
        },
        {
            "name": "sebastian/diff",
            "version": "4.0.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/diff.git",
                "reference": "3461e3fccc7cfdfc2720be910d3bd73c69be590d"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/diff/zipball/3461e3fccc7cfdfc2720be910d3bd73c69be590d",
                "reference": "3461e3fccc7cfdfc2720be910d3bd73c69be590d",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3",
                "symfony/process": "^4.2 || ^5"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                },
                {
                    "name": "Kore Nordmann",
                    "email": "mail@kore-nordmann.de"
                }
            ],
            "description": "Diff implementation",
            "homepage": "https://github.com/sebastianbergmann/diff",
            "keywords": [
                "diff",
                "udiff",
                "unidiff",
                "unified diff"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/diff/issues",
                "source": "https://github.com/sebastianbergmann/diff/tree/4.0.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:10:38+00:00"
        },
        {
            "name": "sebastian/environment",
            "version": "5.1.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/environment.git",
                "reference": "388b6ced16caa751030f6a69e588299fa09200ac"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/environment/zipball/388b6ced16caa751030f6a69e588299fa09200ac",
                "reference": "388b6ced16caa751030f6a69e588299fa09200ac",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "suggest": {
                "ext-posix": "*"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "5.1-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Provides functionality to handle HHVM/PHP environments",
            "homepage": "http://www.github.com/sebastianbergmann/environment",
            "keywords": [
                "Xdebug",
                "environment",
                "hhvm"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/environment/issues",
                "source": "https://github.com/sebastianbergmann/environment/tree/5.1.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T05:52:38+00:00"
        },
        {
            "name": "sebastian/exporter",
            "version": "4.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/exporter.git",
                "reference": "d89cc98761b8cb5a1a235a6b703ae50d34080e65"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/exporter/zipball/d89cc98761b8cb5a1a235a6b703ae50d34080e65",
                "reference": "d89cc98761b8cb5a1a235a6b703ae50d34080e65",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3",
                "sebastian/recursion-context": "^4.0"
            },
            "require-dev": {
                "ext-mbstring": "*",
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                },
                {
                    "name": "Jeff Welch",
                    "email": "whatthejeff@gmail.com"
                },
                {
                    "name": "Volker Dusch",
                    "email": "github@wallbash.com"
                },
                {
                    "name": "Adam Harvey",
                    "email": "aharvey@php.net"
                },
                {
                    "name": "Bernhard Schussek",
                    "email": "bschussek@gmail.com"
                }
            ],
            "description": "Provides the functionality to export PHP variables for visualization",
            "homepage": "http://www.github.com/sebastianbergmann/exporter",
            "keywords": [
                "export",
                "exporter"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/exporter/issues",
                "source": "https://github.com/sebastianbergmann/exporter/tree/4.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T05:24:23+00:00"
        },
        {
            "name": "sebastian/global-state",
            "version": "5.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/global-state.git",
                "reference": "23bd5951f7ff26f12d4e3242864df3e08dec4e49"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/global-state/zipball/23bd5951f7ff26f12d4e3242864df3e08dec4e49",
                "reference": "23bd5951f7ff26f12d4e3242864df3e08dec4e49",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3",
                "sebastian/object-reflector": "^2.0",
                "sebastian/recursion-context": "^4.0"
            },
            "require-dev": {
                "ext-dom": "*",
                "phpunit/phpunit": "^9.3"
            },
            "suggest": {
                "ext-uopz": "*"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "5.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Snapshotting of global state",
            "homepage": "http://www.github.com/sebastianbergmann/global-state",
            "keywords": [
                "global state"
            ],
            "support": {
                "issues": "https://github.com/sebastianbergmann/global-state/issues",
                "source": "https://github.com/sebastianbergmann/global-state/tree/5.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2021-06-11T13:31:12+00:00"
        },
        {
            "name": "sebastian/lines-of-code",
            "version": "1.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/lines-of-code.git",
                "reference": "c1c2e997aa3146983ed888ad08b15470a2e22ecc"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/lines-of-code/zipball/c1c2e997aa3146983ed888ad08b15470a2e22ecc",
                "reference": "c1c2e997aa3146983ed888ad08b15470a2e22ecc",
                "shasum": ""
            },
            "require": {
                "nikic/php-parser": "^4.6",
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "1.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Library for counting the lines of code in PHP source code",
            "homepage": "https://github.com/sebastianbergmann/lines-of-code",
            "support": {
                "issues": "https://github.com/sebastianbergmann/lines-of-code/issues",
                "source": "https://github.com/sebastianbergmann/lines-of-code/tree/1.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-11-28T06:42:11+00:00"
        },
        {
            "name": "sebastian/object-enumerator",
            "version": "4.0.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/object-enumerator.git",
                "reference": "5c9eeac41b290a3712d88851518825ad78f45c71"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/object-enumerator/zipball/5c9eeac41b290a3712d88851518825ad78f45c71",
                "reference": "5c9eeac41b290a3712d88851518825ad78f45c71",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3",
                "sebastian/object-reflector": "^2.0",
                "sebastian/recursion-context": "^4.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Traverses array structures and object graphs to enumerate all referenced objects",
            "homepage": "https://github.com/sebastianbergmann/object-enumerator/",
            "support": {
                "issues": "https://github.com/sebastianbergmann/object-enumerator/issues",
                "source": "https://github.com/sebastianbergmann/object-enumerator/tree/4.0.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:12:34+00:00"
        },
        {
            "name": "sebastian/object-reflector",
            "version": "2.0.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/object-reflector.git",
                "reference": "b4f479ebdbf63ac605d183ece17d8d7fe49c15c7"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/object-reflector/zipball/b4f479ebdbf63ac605d183ece17d8d7fe49c15c7",
                "reference": "b4f479ebdbf63ac605d183ece17d8d7fe49c15c7",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Allows reflection of object attributes, including inherited and non-public ones",
            "homepage": "https://github.com/sebastianbergmann/object-reflector/",
            "support": {
                "issues": "https://github.com/sebastianbergmann/object-reflector/issues",
                "source": "https://github.com/sebastianbergmann/object-reflector/tree/2.0.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:14:26+00:00"
        },
        {
            "name": "sebastian/recursion-context",
            "version": "4.0.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/recursion-context.git",
                "reference": "cd9d8cf3c5804de4341c283ed787f099f5506172"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/recursion-context/zipball/cd9d8cf3c5804de4341c283ed787f099f5506172",
                "reference": "cd9d8cf3c5804de4341c283ed787f099f5506172",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                },
                {
                    "name": "Jeff Welch",
                    "email": "whatthejeff@gmail.com"
                },
                {
                    "name": "Adam Harvey",
                    "email": "aharvey@php.net"
                }
            ],
            "description": "Provides functionality to recursively process PHP variables",
            "homepage": "http://www.github.com/sebastianbergmann/recursion-context",
            "support": {
                "issues": "https://github.com/sebastianbergmann/recursion-context/issues",
                "source": "https://github.com/sebastianbergmann/recursion-context/tree/4.0.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-10-26T13:17:30+00:00"
        },
        {
            "name": "sebastian/resource-operations",
            "version": "3.0.3",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/resource-operations.git",
                "reference": "0f4443cb3a1d92ce809899753bc0d5d5a8dd19a8"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/resource-operations/zipball/0f4443cb3a1d92ce809899753bc0d5d5a8dd19a8",
                "reference": "0f4443cb3a1d92ce809899753bc0d5d5a8dd19a8",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.0"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de"
                }
            ],
            "description": "Provides a list of PHP built-in functions that operate on resources",
            "homepage": "https://www.github.com/sebastianbergmann/resource-operations",
            "support": {
                "issues": "https://github.com/sebastianbergmann/resource-operations/issues",
                "source": "https://github.com/sebastianbergmann/resource-operations/tree/3.0.3"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T06:45:17+00:00"
        },
        {
            "name": "sebastian/type",
            "version": "2.3.4",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/type.git",
                "reference": "b8cd8a1c753c90bc1a0f5372170e3e489136f914"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/type/zipball/b8cd8a1c753c90bc1a0f5372170e3e489136f914",
                "reference": "b8cd8a1c753c90bc1a0f5372170e3e489136f914",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.3-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Collection of value objects that represent the types of the PHP type system",
            "homepage": "https://github.com/sebastianbergmann/type",
            "support": {
                "issues": "https://github.com/sebastianbergmann/type/issues",
                "source": "https://github.com/sebastianbergmann/type/tree/2.3.4"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2021-06-15T12:49:02+00:00"
        },
        {
            "name": "sebastian/version",
            "version": "3.0.2",
            "source": {
                "type": "git",
                "url": "https://github.com/sebastianbergmann/version.git",
                "reference": "c6c1022351a901512170118436c764e473f6de8c"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/sebastianbergmann/version/zipball/c6c1022351a901512170118436c764e473f6de8c",
                "reference": "c6c1022351a901512170118436c764e473f6de8c",
                "shasum": ""
            },
            "require": {
                "php": ">=7.3"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.0-dev"
                }
            },
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Sebastian Bergmann",
                    "email": "sebastian@phpunit.de",
                    "role": "lead"
                }
            ],
            "description": "Library that helps with managing the version number of Git-hosted PHP projects",
            "homepage": "https://github.com/sebastianbergmann/version",
            "support": {
                "issues": "https://github.com/sebastianbergmann/version/issues",
                "source": "https://github.com/sebastianbergmann/version/tree/3.0.2"
            },
            "funding": [
                {
                    "url": "https://github.com/sebastianbergmann",
                    "type": "github"
                }
            ],
            "time": "2020-09-28T06:39:44+00:00"
        },
        {
            "name": "slevomat/coding-standard",
            "version": "7.0.14",
            "source": {
                "type": "git",
                "url": "https://github.com/slevomat/coding-standard.git",
                "reference": "15b2b4630c148775debea8e412bc7e128d9868a3"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/slevomat/coding-standard/zipball/15b2b4630c148775debea8e412bc7e128d9868a3",
                "reference": "15b2b4630c148775debea8e412bc7e128d9868a3",
                "shasum": ""
            },
            "require": {
                "dealerdirect/phpcodesniffer-composer-installer": "^0.6.2 || ^0.7",
                "php": "^7.1 || ^8.0",
                "phpstan/phpdoc-parser": "0.5.1 - 0.5.5",
                "squizlabs/php_codesniffer": "^3.6.0"
            },
            "require-dev": {
                "phing/phing": "2.16.4",
                "php-parallel-lint/php-parallel-lint": "1.3.1",
                "phpstan/phpstan": "0.12.96",
                "phpstan/phpstan-deprecation-rules": "0.12.6",
                "phpstan/phpstan-phpunit": "0.12.22",
                "phpstan/phpstan-strict-rules": "0.12.11",
                "phpunit/phpunit": "7.5.20|8.5.5|9.5.8"
            },
            "type": "phpcodesniffer-standard",
            "extra": {
                "branch-alias": {
                    "dev-master": "7.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "SlevomatCodingStandard\\": "SlevomatCodingStandard"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "description": "Slevomat Coding Standard for PHP_CodeSniffer complements Consistence Coding Standard by providing sniffs with additional checks.",
            "support": {
                "issues": "https://github.com/slevomat/coding-standard/issues",
                "source": "https://github.com/slevomat/coding-standard/tree/7.0.14"
            },
            "funding": [
                {
                    "url": "https://github.com/kukulich",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/slevomat/coding-standard",
                    "type": "tidelift"
                }
            ],
            "time": "2021-08-26T12:17:56+00:00"
        },
        {
            "name": "squizlabs/php_codesniffer",
            "version": "3.6.0",
            "source": {
                "type": "git",
                "url": "https://github.com/squizlabs/PHP_CodeSniffer.git",
                "reference": "ffced0d2c8fa8e6cdc4d695a743271fab6c38625"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/squizlabs/PHP_CodeSniffer/zipball/ffced0d2c8fa8e6cdc4d695a743271fab6c38625",
                "reference": "ffced0d2c8fa8e6cdc4d695a743271fab6c38625",
                "shasum": ""
            },
            "require": {
                "ext-simplexml": "*",
                "ext-tokenizer": "*",
                "ext-xmlwriter": "*",
                "php": ">=5.4.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^4.0 || ^5.0 || ^6.0 || ^7.0"
            },
            "bin": [
                "bin/phpcs",
                "bin/phpcbf"
            ],
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.x-dev"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Greg Sherwood",
                    "role": "lead"
                }
            ],
            "description": "PHP_CodeSniffer tokenizes PHP, JavaScript and CSS files and detects violations of a defined set of coding standards.",
            "homepage": "https://github.com/squizlabs/PHP_CodeSniffer",
            "keywords": [
                "phpcs",
                "standards"
            ],
            "support": {
                "issues": "https://github.com/squizlabs/PHP_CodeSniffer/issues",
                "source": "https://github.com/squizlabs/PHP_CodeSniffer",
                "wiki": "https://github.com/squizlabs/PHP_CodeSniffer/wiki"
            },
            "time": "2021-04-09T00:54:41+00:00"
        },
        {
            "name": "symfony/console",
            "version": "v5.3.7",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/console.git",
                "reference": "8b1008344647462ae6ec57559da166c2bfa5e16a"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/console/zipball/8b1008344647462ae6ec57559da166c2bfa5e16a",
                "reference": "8b1008344647462ae6ec57559da166c2bfa5e16a",
                "shasum": ""
            },
            "require": {
                "php": ">=7.2.5",
                "symfony/deprecation-contracts": "^2.1",
                "symfony/polyfill-mbstring": "~1.0",
                "symfony/polyfill-php73": "^1.8",
                "symfony/polyfill-php80": "^1.16",
                "symfony/service-contracts": "^1.1|^2",
                "symfony/string": "^5.1"
            },
            "conflict": {
                "psr/log": ">=3",
                "symfony/dependency-injection": "<4.4",
                "symfony/dotenv": "<5.1",
                "symfony/event-dispatcher": "<4.4",
                "symfony/lock": "<4.4",
                "symfony/process": "<4.4"
            },
            "provide": {
                "psr/log-implementation": "1.0|2.0"
            },
            "require-dev": {
                "psr/log": "^1|^2",
                "symfony/config": "^4.4|^5.0",
                "symfony/dependency-injection": "^4.4|^5.0",
                "symfony/event-dispatcher": "^4.4|^5.0",
                "symfony/lock": "^4.4|^5.0",
                "symfony/process": "^4.4|^5.0",
                "symfony/var-dumper": "^4.4|^5.0"
            },
            "suggest": {
                "psr/log": "For using the console logger",
                "symfony/event-dispatcher": "",
                "symfony/lock": "",
                "symfony/process": ""
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Symfony\\Component\\Console\\": ""
                },
                "exclude-from-classmap": [
                    "/Tests/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Fabien Potencier",
                    "email": "fabien@symfony.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Eases the creation of beautiful and testable command line interfaces",
            "homepage": "https://symfony.com",
            "keywords": [
                "cli",
                "command line",
                "console",
                "terminal"
            ],
            "support": {
                "source": "https://github.com/symfony/console/tree/v5.3.7"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-08-25T20:02:16+00:00"
        },
        {
            "name": "symfony/deprecation-contracts",
            "version": "v2.4.0",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/deprecation-contracts.git",
                "reference": "5f38c8804a9e97d23e0c8d63341088cd8a22d627"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/deprecation-contracts/zipball/5f38c8804a9e97d23e0c8d63341088cd8a22d627",
                "reference": "5f38c8804a9e97d23e0c8d63341088cd8a22d627",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "2.4-dev"
                },
                "thanks": {
                    "name": "symfony/contracts",
                    "url": "https://github.com/symfony/contracts"
                }
            },
            "autoload": {
                "files": [
                    "function.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "A generic function and convention to trigger deprecation notices",
            "homepage": "https://symfony.com",
            "support": {
                "source": "https://github.com/symfony/deprecation-contracts/tree/v2.4.0"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-03-23T23:28:01+00:00"
        },
        {
            "name": "symfony/polyfill-ctype",
            "version": "v1.23.0",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-ctype.git",
                "reference": "46cd95797e9df938fdd2b03693b5fca5e64b01ce"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-ctype/zipball/46cd95797e9df938fdd2b03693b5fca5e64b01ce",
                "reference": "46cd95797e9df938fdd2b03693b5fca5e64b01ce",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "suggest": {
                "ext-ctype": "For best performance"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Ctype\\": ""
                },
                "files": [
                    "bootstrap.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Gert de Pagter",
                    "email": "BackEndTea@gmail.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill for ctype functions",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "ctype",
                "polyfill",
                "portable"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-ctype/tree/v1.23.0"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-02-19T12:13:01+00:00"
        },
        {
            "name": "symfony/polyfill-intl-grapheme",
            "version": "v1.23.1",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-intl-grapheme.git",
                "reference": "16880ba9c5ebe3642d1995ab866db29270b36535"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-intl-grapheme/zipball/16880ba9c5ebe3642d1995ab866db29270b36535",
                "reference": "16880ba9c5ebe3642d1995ab866db29270b36535",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "suggest": {
                "ext-intl": "For best performance"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Intl\\Grapheme\\": ""
                },
                "files": [
                    "bootstrap.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill for intl's grapheme_* functions",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "grapheme",
                "intl",
                "polyfill",
                "portable",
                "shim"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-intl-grapheme/tree/v1.23.1"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-05-27T12:26:48+00:00"
        },
        {
            "name": "symfony/polyfill-intl-normalizer",
            "version": "v1.23.0",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-intl-normalizer.git",
                "reference": "8590a5f561694770bdcd3f9b5c69dde6945028e8"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-intl-normalizer/zipball/8590a5f561694770bdcd3f9b5c69dde6945028e8",
                "reference": "8590a5f561694770bdcd3f9b5c69dde6945028e8",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "suggest": {
                "ext-intl": "For best performance"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Intl\\Normalizer\\": ""
                },
                "files": [
                    "bootstrap.php"
                ],
                "classmap": [
                    "Resources/stubs"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill for intl's Normalizer class and related functions",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "intl",
                "normalizer",
                "polyfill",
                "portable",
                "shim"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-intl-normalizer/tree/v1.23.0"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-02-19T12:13:01+00:00"
        },
        {
            "name": "symfony/polyfill-mbstring",
            "version": "v1.23.1",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-mbstring.git",
                "reference": "9174a3d80210dca8daa7f31fec659150bbeabfc6"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-mbstring/zipball/9174a3d80210dca8daa7f31fec659150bbeabfc6",
                "reference": "9174a3d80210dca8daa7f31fec659150bbeabfc6",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "suggest": {
                "ext-mbstring": "For best performance"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Mbstring\\": ""
                },
                "files": [
                    "bootstrap.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill for the Mbstring extension",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "mbstring",
                "polyfill",
                "portable",
                "shim"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-mbstring/tree/v1.23.1"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-05-27T12:26:48+00:00"
        },
        {
            "name": "symfony/polyfill-php73",
            "version": "v1.23.0",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-php73.git",
                "reference": "fba8933c384d6476ab14fb7b8526e5287ca7e010"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-php73/zipball/fba8933c384d6476ab14fb7b8526e5287ca7e010",
                "reference": "fba8933c384d6476ab14fb7b8526e5287ca7e010",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Php73\\": ""
                },
                "files": [
                    "bootstrap.php"
                ],
                "classmap": [
                    "Resources/stubs"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill backporting some PHP 7.3+ features to lower PHP versions",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "polyfill",
                "portable",
                "shim"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-php73/tree/v1.23.0"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-02-19T12:13:01+00:00"
        },
        {
            "name": "symfony/polyfill-php80",
            "version": "v1.23.1",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/polyfill-php80.git",
                "reference": "1100343ed1a92e3a38f9ae122fc0eb21602547be"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/polyfill-php80/zipball/1100343ed1a92e3a38f9ae122fc0eb21602547be",
                "reference": "1100343ed1a92e3a38f9ae122fc0eb21602547be",
                "shasum": ""
            },
            "require": {
                "php": ">=7.1"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "1.23-dev"
                },
                "thanks": {
                    "name": "symfony/polyfill",
                    "url": "https://github.com/symfony/polyfill"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Polyfill\\Php80\\": ""
                },
                "files": [
                    "bootstrap.php"
                ],
                "classmap": [
                    "Resources/stubs"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Ion Bazan",
                    "email": "ion.bazan@gmail.com"
                },
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Symfony polyfill backporting some PHP 8.0+ features to lower PHP versions",
            "homepage": "https://symfony.com",
            "keywords": [
                "compatibility",
                "polyfill",
                "portable",
                "shim"
            ],
            "support": {
                "source": "https://github.com/symfony/polyfill-php80/tree/v1.23.1"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-07-28T13:41:28+00:00"
        },
        {
            "name": "symfony/service-contracts",
            "version": "v2.4.0",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/service-contracts.git",
                "reference": "f040a30e04b57fbcc9c6cbcf4dbaa96bd318b9bb"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/service-contracts/zipball/f040a30e04b57fbcc9c6cbcf4dbaa96bd318b9bb",
                "reference": "f040a30e04b57fbcc9c6cbcf4dbaa96bd318b9bb",
                "shasum": ""
            },
            "require": {
                "php": ">=7.2.5",
                "psr/container": "^1.1"
            },
            "suggest": {
                "symfony/service-implementation": ""
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-main": "2.4-dev"
                },
                "thanks": {
                    "name": "symfony/contracts",
                    "url": "https://github.com/symfony/contracts"
                }
            },
            "autoload": {
                "psr-4": {
                    "Symfony\\Contracts\\Service\\": ""
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Generic abstractions related to writing services",
            "homepage": "https://symfony.com",
            "keywords": [
                "abstractions",
                "contracts",
                "decoupling",
                "interfaces",
                "interoperability",
                "standards"
            ],
            "support": {
                "source": "https://github.com/symfony/service-contracts/tree/v2.4.0"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-04-01T10:43:52+00:00"
        },
        {
            "name": "symfony/string",
            "version": "v5.3.7",
            "source": {
                "type": "git",
                "url": "https://github.com/symfony/string.git",
                "reference": "8d224396e28d30f81969f083a58763b8b9ceb0a5"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/symfony/string/zipball/8d224396e28d30f81969f083a58763b8b9ceb0a5",
                "reference": "8d224396e28d30f81969f083a58763b8b9ceb0a5",
                "shasum": ""
            },
            "require": {
                "php": ">=7.2.5",
                "symfony/polyfill-ctype": "~1.8",
                "symfony/polyfill-intl-grapheme": "~1.0",
                "symfony/polyfill-intl-normalizer": "~1.0",
                "symfony/polyfill-mbstring": "~1.0",
                "symfony/polyfill-php80": "~1.15"
            },
            "require-dev": {
                "symfony/error-handler": "^4.4|^5.0",
                "symfony/http-client": "^4.4|^5.0",
                "symfony/translation-contracts": "^1.1|^2",
                "symfony/var-exporter": "^4.4|^5.0"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Symfony\\Component\\String\\": ""
                },
                "files": [
                    "Resources/functions.php"
                ],
                "exclude-from-classmap": [
                    "/Tests/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Nicolas Grekas",
                    "email": "p@tchwork.com"
                },
                {
                    "name": "Symfony Community",
                    "homepage": "https://symfony.com/contributors"
                }
            ],
            "description": "Provides an object-oriented API to strings and deals with bytes, UTF-8 code points and grapheme clusters in a unified way",
            "homepage": "https://symfony.com",
            "keywords": [
                "grapheme",
                "i18n",
                "string",
                "unicode",
                "utf-8",
                "utf8"
            ],
            "support": {
                "source": "https://github.com/symfony/string/tree/v5.3.7"
            },
            "funding": [
                {
                    "url": "https://symfony.com/sponsor",
                    "type": "custom"
                },
                {
                    "url": "https://github.com/fabpot",
                    "type": "github"
                },
                {
                    "url": "https://tidelift.com/funding/github/packagist/symfony/symfony",
                    "type": "tidelift"
                }
            ],
            "time": "2021-08-26T08:00:08+00:00"
        },
        {
            "name": "theseer/tokenizer",
            "version": "1.2.1",
            "source": {
                "type": "git",
                "url": "https://github.com/theseer/tokenizer.git",
                "reference": "34a41e998c2183e22995f158c581e7b5e755ab9e"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/theseer/tokenizer/zipball/34a41e998c2183e22995f158c581e7b5e755ab9e",
                "reference": "34a41e998c2183e22995f158c581e7b5e755ab9e",
                "shasum": ""
            },
            "require": {
                "ext-dom": "*",
                "ext-tokenizer": "*",
                "ext-xmlwriter": "*",
                "php": "^7.2 || ^8.0"
            },
            "type": "library",
            "autoload": {
                "classmap": [
                    "src/"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-3-Clause"
            ],
            "authors": [
                {
                    "name": "Arne Blankerts",
                    "email": "arne@blankerts.de",
                    "role": "Developer"
                }
            ],
            "description": "A small library for converting tokenized PHP source code into XML and potentially other formats",
            "support": {
                "issues": "https://github.com/theseer/tokenizer/issues",
                "source": "https://github.com/theseer/tokenizer/tree/1.2.1"
            },
            "funding": [
                {
                    "url": "https://github.com/theseer",
                    "type": "github"
                }
            ],
            "time": "2021-07-28T10:34:58+00:00"
        },
        {
            "name": "vimeo/psalm",
            "version": "3.18.2",
            "source": {
                "type": "git",
                "url": "https://github.com/vimeo/psalm.git",
                "reference": "19aa905f7c3c7350569999a93c40ae91ae4e1626"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/vimeo/psalm/zipball/19aa905f7c3c7350569999a93c40ae91ae4e1626",
                "reference": "19aa905f7c3c7350569999a93c40ae91ae4e1626",
                "shasum": ""
            },
            "require": {
                "amphp/amp": "^2.1",
                "amphp/byte-stream": "^1.5",
                "composer/package-versions-deprecated": "^1.8.0",
                "composer/semver": "^1.4 || ^2.0 || ^3.0",
                "composer/xdebug-handler": "^1.1",
                "dnoegel/php-xdg-base-dir": "^0.1.1",
                "ext-dom": "*",
                "ext-json": "*",
                "ext-libxml": "*",
                "ext-mbstring": "*",
                "ext-simplexml": "*",
                "ext-tokenizer": "*",
                "felixfbecker/advanced-json-rpc": "^3.0.3",
                "felixfbecker/language-server-protocol": "^1.4",
                "netresearch/jsonmapper": "^1.0 || ^2.0 || ^3.0",
                "nikic/php-parser": "4.3.* || 4.4.* || 4.5.* || 4.6.* || ^4.8",
                "openlss/lib-array2xml": "^1.0",
                "php": "^7.1.3|^8",
                "sebastian/diff": "^3.0 || ^4.0",
                "symfony/console": "^3.4.17 || ^4.1.6 || ^5.0",
                "webmozart/glob": "^4.1",
                "webmozart/path-util": "^2.3"
            },
            "provide": {
                "psalm/psalm": "self.version"
            },
            "require-dev": {
                "amphp/amp": "^2.4.2",
                "bamarni/composer-bin-plugin": "^1.2",
                "brianium/paratest": "^4.0.0",
                "ext-curl": "*",
                "phpdocumentor/reflection-docblock": "^4.3.4 || ^5",
                "phpmyadmin/sql-parser": "5.1.0",
                "phpspec/prophecy": ">=1.9.0",
                "phpunit/phpunit": "^7.5.16 || ^8.5 || ^9.0",
                "psalm/plugin-phpunit": "^0.11",
                "slevomat/coding-standard": "^5.0",
                "squizlabs/php_codesniffer": "^3.5",
                "symfony/process": "^4.3",
                "weirdan/prophecy-shim": "^1.0 || ^2.0"
            },
            "suggest": {
                "ext-igbinary": "^2.0.5"
            },
            "bin": [
                "psalm",
                "psalm-language-server",
                "psalm-plugin",
                "psalm-refactor",
                "psalter"
            ],
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "3.x-dev",
                    "dev-2.x": "2.x-dev",
                    "dev-1.x": "1.x-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Psalm\\": "src/Psalm/"
                },
                "files": [
                    "src/functions.php",
                    "src/spl_object_id.php"
                ]
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Matthew Brown"
                }
            ],
            "description": "A static analysis tool for finding errors in PHP applications",
            "keywords": [
                "code",
                "inspection",
                "php"
            ],
            "support": {
                "issues": "https://github.com/vimeo/psalm/issues",
                "source": "https://github.com/vimeo/psalm/tree/3.18.2"
            },
            "time": "2020-10-20T13:48:22+00:00"
        },
        {
            "name": "webimpress/coding-standard",
            "version": "1.2.2",
            "source": {
                "type": "git",
                "url": "https://github.com/webimpress/coding-standard.git",
                "reference": "8f4a220de33f471a8101836f7ec72b852c3f9f03"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/webimpress/coding-standard/zipball/8f4a220de33f471a8101836f7ec72b852c3f9f03",
                "reference": "8f4a220de33f471a8101836f7ec72b852c3f9f03",
                "shasum": ""
            },
            "require": {
                "php": "^7.3 || ^8.0",
                "squizlabs/php_codesniffer": "^3.6"
            },
            "require-dev": {
                "phpunit/phpunit": "^9.5.4"
            },
            "type": "phpcodesniffer-standard",
            "extra": {
                "dev-master": "1.2.x-dev",
                "dev-develop": "1.3.x-dev"
            },
            "autoload": {
                "psr-4": {
                    "WebimpressCodingStandard\\": "src/WebimpressCodingStandard/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "BSD-2-Clause"
            ],
            "description": "Webimpress Coding Standard",
            "keywords": [
                "Coding Standard",
                "PSR-2",
                "phpcs",
                "psr-12",
                "webimpress"
            ],
            "support": {
                "issues": "https://github.com/webimpress/coding-standard/issues",
                "source": "https://github.com/webimpress/coding-standard/tree/1.2.2"
            },
            "funding": [
                {
                    "url": "https://github.com/michalbundyra",
                    "type": "github"
                }
            ],
            "time": "2021-04-12T12:51:27+00:00"
        },
        {
            "name": "webmozart/assert",
            "version": "1.9.1",
            "source": {
                "type": "git",
                "url": "https://github.com/webmozarts/assert.git",
                "reference": "bafc69caeb4d49c39fd0779086c03a3738cbb389"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/webmozarts/assert/zipball/bafc69caeb4d49c39fd0779086c03a3738cbb389",
                "reference": "bafc69caeb4d49c39fd0779086c03a3738cbb389",
                "shasum": ""
            },
            "require": {
                "php": "^5.3.3 || ^7.0 || ^8.0",
                "symfony/polyfill-ctype": "^1.8"
            },
            "conflict": {
                "phpstan/phpstan": "<0.12.20",
                "vimeo/psalm": "<3.9.1"
            },
            "require-dev": {
                "phpunit/phpunit": "^4.8.36 || ^7.5.13"
            },
            "type": "library",
            "autoload": {
                "psr-4": {
                    "Webmozart\\Assert\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Bernhard Schussek",
                    "email": "bschussek@gmail.com"
                }
            ],
            "description": "Assertions to validate method input/output with nice error messages.",
            "keywords": [
                "assert",
                "check",
                "validate"
            ],
            "support": {
                "issues": "https://github.com/webmozarts/assert/issues",
                "source": "https://github.com/webmozarts/assert/tree/1.9.1"
            },
            "time": "2020-07-08T17:02:28+00:00"
        },
        {
            "name": "webmozart/glob",
            "version": "4.3.0",
            "source": {
                "type": "git",
                "url": "https://github.com/webmozarts/glob.git",
                "reference": "06358fafde0f32edb4513f4fd88fe113a40c90ee"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/webmozarts/glob/zipball/06358fafde0f32edb4513f4fd88fe113a40c90ee",
                "reference": "06358fafde0f32edb4513f4fd88fe113a40c90ee",
                "shasum": ""
            },
            "require": {
                "php": "^7.3 || ^8.0.0",
                "webmozart/path-util": "^2.2"
            },
            "require-dev": {
                "phpunit/phpunit": "^8.0",
                "symfony/filesystem": "^5.1"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "4.1-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Webmozart\\Glob\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Bernhard Schussek",
                    "email": "bschussek@gmail.com"
                }
            ],
            "description": "A PHP implementation of Ant's glob.",
            "support": {
                "issues": "https://github.com/webmozarts/glob/issues",
                "source": "https://github.com/webmozarts/glob/tree/4.3.0"
            },
            "time": "2021-01-21T06:17:15+00:00"
        },
        {
            "name": "webmozart/path-util",
            "version": "2.3.0",
            "source": {
                "type": "git",
                "url": "https://github.com/webmozart/path-util.git",
                "reference": "d939f7edc24c9a1bb9c0dee5cb05d8e859490725"
            },
            "dist": {
                "type": "zip",
                "url": "https://api.github.com/repos/webmozart/path-util/zipball/d939f7edc24c9a1bb9c0dee5cb05d8e859490725",
                "reference": "d939f7edc24c9a1bb9c0dee5cb05d8e859490725",
                "shasum": ""
            },
            "require": {
                "php": ">=5.3.3",
                "webmozart/assert": "~1.0"
            },
            "require-dev": {
                "phpunit/phpunit": "^4.6",
                "sebastian/version": "^1.0.1"
            },
            "type": "library",
            "extra": {
                "branch-alias": {
                    "dev-master": "2.3-dev"
                }
            },
            "autoload": {
                "psr-4": {
                    "Webmozart\\PathUtil\\": "src/"
                }
            },
            "notification-url": "https://packagist.org/downloads/",
            "license": [
                "MIT"
            ],
            "authors": [
                {
                    "name": "Bernhard Schussek",
                    "email": "bschussek@gmail.com"
                }
            ],
            "description": "A robust cross-platform utility for normalizing, comparing and modifying file paths.",
            "support": {
                "issues": "https://github.com/webmozart/path-util/issues",
                "source": "https://github.com/webmozart/path-util/tree/2.3.0"
            },
            "time": "2015-12-17T08:42:14+00:00"
        }
    ],
    "aliases": [],
    "minimum-stability": "stable",
    "stability-flags": [],
    "prefer-stable": false,
    "prefer-lowest": false,
    "platform": {
        "php": "^7.3 || ~8.0.0 || ~8.1.0"
    },
    "platform-dev": [],
    "plugin-api-version": "2.0.0"
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/COPYRIGHT.md

Copyright (c) 2020 Laminas Project a Series of LF Projects, LLC. (https://getlaminas.org/)

## File: word/PHP_Word/vendor/laminas/laminas-escaper/LICENSE.md

Copyright (c) 2020 Laminas Project a Series of LF Projects, LLC.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

- Neither the name of Laminas Foundation nor the names of its contributors may
  be used to endorse or promote products derived from this software without
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## File: word/PHP_Word/vendor/laminas/laminas-escaper/phpcs.xml.dist

<?xml version="1.0"?>
<ruleset
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="./vendor/squizlabs/php_codesniffer/phpcs.xsd">

    <arg name="basepath" value="."/>
    <arg name="cache" value=".phpcs-cache"/>
    <arg name="colors"/>
    <arg name="extensions" value="php"/>
    <arg name="parallel" value="80"/>

    <!-- Show progress -->
    <arg value="p"/>

    <!-- Paths to check -->
    <file>src</file>
    <file>test</file>

    <!-- Include all rules from Laminas Coding Standard -->
    <rule ref="LaminasCodingStandard"/>
</ruleset>

## File: word/PHP_Word/vendor/laminas/laminas-escaper/psalm-baseline.xml

<?xml version="1.0" encoding="UTF-8"?>
<files psalm-version="3.18.2@19aa905f7c3c7350569999a93c40ae91ae4e1626">
  <file src="src/Escaper.php">
    <MixedArgument occurrences="10">
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
    </MixedArgument>
    <MixedArgumentTypeCoercion occurrences="3">
      <code>$this-&gt;cssMatcher</code>
      <code>$this-&gt;htmlAttrMatcher</code>
      <code>$this-&gt;jsMatcher</code>
    </MixedArgumentTypeCoercion>
    <MixedAssignment occurrences="3">
      <code>$chr</code>
      <code>$chr</code>
      <code>$chr</code>
    </MixedAssignment>
    <MixedOperand occurrences="1">
      <code>static::$htmlNamedEntityMap[$ord]</code>
    </MixedOperand>
    <PossiblyInvalidArgument occurrences="1">
      <code>$from</code>
    </PossiblyInvalidArgument>
  </file>
  <file src="test/EscaperTest.php">
    <InvalidReturnStatement occurrences="5"/>
    <InvalidReturnType occurrences="5">
      <code>array&lt;string, array{0: string, 1: string}&gt;</code>
      <code>array&lt;string, array{0: string, 1: string}&gt;</code>
      <code>array&lt;string, array{0: string, 1: string}&gt;</code>
      <code>array&lt;string, array{0: string, 1: string}&gt;</code>
      <code>array&lt;string, array{0: string}&gt;</code>
    </InvalidReturnType>
    <MissingReturnType occurrences="9">
      <code>testCssEscapingReturnsStringIfContainsOnlyDigits</code>
      <code>testCssEscapingReturnsStringIfZeroLength</code>
      <code>testHtmlAttributeEscapingEscapesOwaspRecommendedRanges</code>
      <code>testJavascriptEscapingReturnsStringIfContainsOnlyDigits</code>
      <code>testJavascriptEscapingReturnsStringIfZeroLength</code>
      <code>testReturnsEncodingFromGetter</code>
      <code>testSettingEncodingToEmptyStringShouldThrowException</code>
      <code>testSettingEncodingToInvalidValueShouldThrowException</code>
      <code>testUnicodeCodepointConversionToUtf8</code>
    </MissingReturnType>
  </file>
  <file src="vendor/symfony/polyfill-mbstring/bootstrap80.php">
    <ParseError occurrences="1">
      <code>=</code>
    </ParseError>
  </file>
</files>

## File: word/PHP_Word/vendor/laminas/laminas-escaper/psalm.xml

<?xml version="1.0"?>
<psalm
    totallyTyped="true"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="https://getpsalm.org/schema/config"
    xsi:schemaLocation="https://getpsalm.org/schema/config vendor/vimeo/psalm/config.xsd"
    errorBaseline="psalm-baseline.xml"
>
    <projectFiles>
        <directory name="src"/>
        <directory name="test"/>
        <ignoreFiles>
            <directory name="vendor"/>
        </ignoreFiles>
    </projectFiles>

    <issueHandlers>
        <InternalMethod>
            <errorLevel type="suppress">
                <referencedMethod name="PHPUnit\Framework\MockObject\Builder\InvocationMocker::method"/>
            </errorLevel>
            <errorLevel type="suppress">
                <referencedMethod name="PHPUnit\Framework\MockObject\Builder\InvocationMocker::willReturn"/>
            </errorLevel>
            <errorLevel type="suppress">
                <referencedMethod name="PHPUnit\Framework\MockObject\Builder\InvocationMocker::with"/>
            </errorLevel>
        </InternalMethod>
    </issueHandlers>

    <plugins>
        <pluginClass class="Psalm\PhpUnitPlugin\Plugin"/>
    </plugins>
</psalm>

## File: word/PHP_Word/vendor/laminas/laminas-escaper/README.md

# laminas-escaper

[![Build Status](https://github.com/laminas/laminas-escaper/actions/workflows/continuous-integration.yml/badge.svg)](https://github.com/laminas/laminas-escaper/actions/workflows/continuous-integration.yml)
[![Coverage Status](https://coveralls.io/repos/github/laminas/laminas-escaper/badge.svg?branch=master)](https://coveralls.io/github/laminas/laminas-escaper?branch=master)

The OWASP Top 10 web security risks study lists Cross-Site Scripting (XSS) in
second place. PHP’s sole functionality against XSS is limited to two functions
of which one is commonly misapplied. Thus, the laminas-escaper component was written.
It offers developers a way to escape output and defend from XSS and related
vulnerabilities by introducing contextual escaping based on peer-reviewed rules.

## Installation

Run the following to install this library:

```bash
$ composer require laminas/laminas-escaper
```

## Documentation

Browse the documentation online at https://docs.laminas.dev/laminas-escaper/

## Support

* [Issues](https://github.com/laminas/laminas-escaper/issues/)
* [Chat](https://laminas.dev/chat/)
* [Forum](https://discourse.laminas.dev/)

## File: word/PHP_Word/vendor/laminas/laminas-escaper/src/Escaper.php

<?php

declare(strict_types=1);

namespace Laminas\Escaper;

use function bin2hex;
use function ctype_digit;
use function function_exists;
use function hexdec;
use function htmlspecialchars;
use function iconv;
use function in_array;
use function mb_convert_encoding;
use function ord;
use function preg_match;
use function preg_replace_callback;
use function rawurlencode;
use function sprintf;
use function strlen;
use function strtolower;
use function strtoupper;
use function substr;

use const ENT_QUOTES;
use const ENT_SUBSTITUTE;

/**
 * Context specific methods for use in secure output escaping
 */
class Escaper
{
    /**
     * Entity Map mapping Unicode codepoints to any available named HTML entities.
     *
     * While HTML supports far more named entities, the lowest common denominator
     * has become HTML5's XML Serialisation which is restricted to the those named
     * entities that XML supports. Using HTML entities would result in this error:
     *     XML Parsing Error: undefined entity
     *
     * @var array
     */
    protected static $htmlNamedEntityMap = [
        34 => 'quot', // quotation mark
        38 => 'amp', // ampersand
        60 => 'lt', // less-than sign
        62 => 'gt', // greater-than sign
    ];

    /**
     * Current encoding for escaping. If not UTF-8, we convert strings from this encoding
     * pre-escaping and back to this encoding post-escaping.
     *
     * @var string
     */
    protected $encoding = 'utf-8';

    /**
     * Holds the value of the special flags passed as second parameter to
     * htmlspecialchars().
     *
     * @var int
     */
    protected $htmlSpecialCharsFlags;

    /**
     * Static Matcher which escapes characters for HTML Attribute contexts
     *
     * @var callable
     */
    protected $htmlAttrMatcher;

    /**
     * Static Matcher which escapes characters for Javascript contexts
     *
     * @var callable
     */
    protected $jsMatcher;

    /**
     * Static Matcher which escapes characters for CSS Attribute contexts
     *
     * @var callable
     */
    protected $cssMatcher;

    /**
     * List of all encoding supported by this class
     *
     * @var array
     */
    protected $supportedEncodings = [
        'iso-8859-1',
        'iso8859-1',
        'iso-8859-5',
        'iso8859-5',
        'iso-8859-15',
        'iso8859-15',
        'utf-8',
        'cp866',
        'ibm866',
        '866',
        'cp1251',
        'windows-1251',
        'win-1251',
        '1251',
        'cp1252',
        'windows-1252',
        '1252',
        'koi8-r',
        'koi8-ru',
        'koi8r',
        'big5',
        '950',
        'gb2312',
        '936',
        'big5-hkscs',
        'shift_jis',
        'sjis',
        'sjis-win',
        'cp932',
        '932',
        'euc-jp',
        'eucjp',
        'eucjp-win',
        'macroman',
    ];

    /**
     * Constructor: Single parameter allows setting of global encoding for use by
     * the current object.
     *
     * @throws Exception\InvalidArgumentException
     */
    public function __construct(?string $encoding = null)
    {
        if ($encoding !== null) {
            if ($encoding === '') {
                throw new Exception\InvalidArgumentException(
                    static::class . ' constructor parameter does not allow a blank value'
                );
            }

            $encoding = strtolower($encoding);
            if (! in_array($encoding, $this->supportedEncodings)) {
                throw new Exception\InvalidArgumentException(
                    'Value of \'' . $encoding . '\' passed to ' . static::class
                    . ' constructor parameter is invalid. Provide an encoding supported by htmlspecialchars()'
                );
            }

            $this->encoding = $encoding;
        }

        // We take advantage of ENT_SUBSTITUTE flag to correctly deal with invalid UTF-8 sequences.
        $this->htmlSpecialCharsFlags = ENT_QUOTES | ENT_SUBSTITUTE;

        // set matcher callbacks
        $this->htmlAttrMatcher = [$this, 'htmlAttrMatcher'];
        $this->jsMatcher       = [$this, 'jsMatcher'];
        $this->cssMatcher      = [$this, 'cssMatcher'];
    }

    /**
     * Return the encoding that all output/input is expected to be encoded in.
     *
     * @return string
     */
    public function getEncoding()
    {
        return $this->encoding;
    }

    /**
     * Escape a string for the HTML Body context where there are very few characters
     * of special meaning. Internally this will use htmlspecialchars().
     *
     * @return string
     */
    public function escapeHtml(string $string)
    {
        return htmlspecialchars($string, $this->htmlSpecialCharsFlags, $this->encoding);
    }

    /**
     * Escape a string for the HTML Attribute context. We use an extended set of characters
     * to escape that are not covered by htmlspecialchars() to cover cases where an attribute
     * might be unquoted or quoted illegally (e.g. backticks are valid quotes for IE).
     *
     * @return string
     */
    public function escapeHtmlAttr(string $string)
    {
        $string = $this->toUtf8($string);
        if ($string === '' || ctype_digit($string)) {
            return $string;
        }

        $result = preg_replace_callback('/[^a-z0-9,\.\-_]/iSu', $this->htmlAttrMatcher, $string);
        return $this->fromUtf8($result);
    }

    /**
     * Escape a string for the Javascript context. This does not use json_encode(). An extended
     * set of characters are escaped beyond ECMAScript's rules for Javascript literal string
     * escaping in order to prevent misinterpretation of Javascript as HTML leading to the
     * injection of special characters and entities. The escaping used should be tolerant
     * of cases where HTML escaping was not applied on top of Javascript escaping correctly.
     * Backslash escaping is not used as it still leaves the escaped character as-is and so
     * is not useful in a HTML context.
     *
     * @return string
     */
    public function escapeJs(string $string)
    {
        $string = $this->toUtf8($string);
        if ($string === '' || ctype_digit($string)) {
            return $string;
        }

        $result = preg_replace_callback('/[^a-z0-9,\._]/iSu', $this->jsMatcher, $string);
        return $this->fromUtf8($result);
    }

    /**
     * Escape a string for the URI or Parameter contexts. This should not be used to escape
     * an entire URI - only a subcomponent being inserted. The function is a simple proxy
     * to rawurlencode() which now implements RFC 3986 since PHP 5.3 completely.
     *
     * @return string
     */
    public function escapeUrl(string $string)
    {
        return rawurlencode($string);
    }

    /**
     * Escape a string for the CSS context. CSS escaping can be applied to any string being
     * inserted into CSS and escapes everything except alphanumerics.
     *
     * @return string
     */
    public function escapeCss(string $string)
    {
        $string = $this->toUtf8($string);
        if ($string === '' || ctype_digit($string)) {
            return $string;
        }

        $result = preg_replace_callback('/[^a-z0-9]/iSu', $this->cssMatcher, $string);
        return $this->fromUtf8($result);
    }

    /**
     * Callback function for preg_replace_callback that applies HTML Attribute
     * escaping to all matches.
     *
     * @param array $matches
     * @return string
     */
    protected function htmlAttrMatcher($matches)
    {
        $chr = $matches[0];
        $ord = ord($chr);

        /**
         * The following replaces characters undefined in HTML with the
         * hex entity for the Unicode replacement character.
         */
        if (
            ($ord <= 0x1f && $chr !== "\t" && $chr !== "\n" && $chr !== "\r")
            || ($ord >= 0x7f && $ord <= 0x9f)
        ) {
            return '&#xFFFD;';
        }

        /**
         * Check if the current character to escape has a name entity we should
         * replace it with while grabbing the integer value of the character.
         */
        if (strlen($chr) > 1) {
            $chr = $this->convertEncoding($chr, 'UTF-32BE', 'UTF-8');
        }

        $hex = bin2hex($chr);
        $ord = hexdec($hex);
        if (isset(static::$htmlNamedEntityMap[$ord])) {
            return '&' . static::$htmlNamedEntityMap[$ord] . ';';
        }

        /**
         * Per OWASP recommendations, we'll use upper hex entities
         * for any other characters where a named entity does not exist.
         */
        if ($ord > 255) {
            return sprintf('&#x%04X;', $ord);
        }
        return sprintf('&#x%02X;', $ord);
    }

    /**
     * Callback function for preg_replace_callback that applies Javascript
     * escaping to all matches.
     *
     * @param array $matches
     * @return string
     */
    protected function jsMatcher($matches)
    {
        $chr = $matches[0];
        if (strlen($chr) === 1) {
            return sprintf('\\x%02X', ord($chr));
        }
        $chr = $this->convertEncoding($chr, 'UTF-16BE', 'UTF-8');
        $hex = strtoupper(bin2hex($chr));
        if (strlen($hex) <= 4) {
            return sprintf('\\u%04s', $hex);
        }
        $highSurrogate = substr($hex, 0, 4);
        $lowSurrogate  = substr($hex, 4, 4);
        return sprintf('\\u%04s\\u%04s', $highSurrogate, $lowSurrogate);
    }

    /**
     * Callback function for preg_replace_callback that applies CSS
     * escaping to all matches.
     *
     * @param array $matches
     * @return string
     */
    protected function cssMatcher($matches)
    {
        $chr = $matches[0];
        if (strlen($chr) === 1) {
            $ord = ord($chr);
        } else {
            $chr = $this->convertEncoding($chr, 'UTF-32BE', 'UTF-8');
            $ord = hexdec(bin2hex($chr));
        }
        return sprintf('\\%X ', $ord);
    }

    /**
     * Converts a string to UTF-8 from the base encoding. The base encoding is set via this
     *
     * @param string $string
     * @throws Exception\RuntimeException
     * @return string
     */
    protected function toUtf8($string)
    {
        if ($this->getEncoding() === 'utf-8') {
            $result = $string;
        } else {
            $result = $this->convertEncoding($string, 'UTF-8', $this->getEncoding());
        }

        if (! $this->isUtf8($result)) {
            throw new Exception\RuntimeException(
                sprintf('String to be escaped was not valid UTF-8 or could not be converted: %s', $result)
            );
        }

        return $result;
    }

    /**
     * Converts a string from UTF-8 to the base encoding. The base encoding is set via this
     *
     * @param string $string
     * @return string
     */
    protected function fromUtf8($string)
    {
        if ($this->getEncoding() === 'utf-8') {
            return $string;
        }

        return $this->convertEncoding($string, $this->getEncoding(), 'UTF-8');
    }

    /**
     * Checks if a given string appears to be valid UTF-8 or not.
     *
     * @param string $string
     * @return bool
     */
    protected function isUtf8($string)
    {
        return $string === '' || preg_match('/^./su', $string);
    }

    /**
     * Encoding conversion helper which wraps iconv and mbstring where they exist or throws
     * and exception where neither is available.
     *
     * @param string $string
     * @param string $to
     * @param array|string $from
     * @throws Exception\RuntimeException
     * @return string
     */
    protected function convertEncoding($string, $to, $from)
    {
        if (function_exists('iconv')) {
            $result = iconv($from, $to, $string);
        } elseif (function_exists('mb_convert_encoding')) {
            $result = mb_convert_encoding($string, $to, $from);
        } else {
            throw new Exception\RuntimeException(
                static::class
                . ' requires either the iconv or mbstring extension to be installed'
                . ' when escaping for non UTF-8 strings.'
            );
        }

        if ($result === false) {
            return ''; // return non-fatal blank string on encoding errors from users
        }
        return $result;
    }
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/src/Exception/ExceptionInterface.php

<?php

declare(strict_types=1);

namespace Laminas\Escaper\Exception;

interface ExceptionInterface
{
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/src/Exception/InvalidArgumentException.php

<?php

declare(strict_types=1);

namespace Laminas\Escaper\Exception;

/**
 * Invalid argument exception
 */
class InvalidArgumentException extends \InvalidArgumentException implements
    ExceptionInterface
{
}

## File: word/PHP_Word/vendor/laminas/laminas-escaper/src/Exception/RuntimeException.php

<?php

declare(strict_types=1);

namespace Laminas\Escaper\Exception;

/**
 * Invalid argument exception
 */
class RuntimeException extends \RuntimeException implements
    ExceptionInterface
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/dependabot.yml

version: 2
updates:
- package-ecosystem: composer
  directory: "/"
  schedule:
    interval: monthly
    time: "11:00"
  open-pull-requests-limit: 10

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/ISSUE_TEMPLATE/bug_report.md

---
name: Bug report
about: Create a report to help improve PHPWord
labels: Bug Report

---

### Describe the Bug

A clear and concise description of what the bug is.

### Steps to Reproduce

Please provide a code sample that reproduces the issue.

```php
<?php
require __DIR__ . '/vendor/autoload.php';

$phpWord = new \PhpOffice\PhpWord\PhpWord();
$section = $phpWord->addSection();
$section->...
```

### Expected Behavior

A clear and concise description of what you expected to happen.

### Current Behavior

What is the current behavior?

### Context

Please fill in your environment information:

- PHP Version:
- PHPWord Version:

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/ISSUE_TEMPLATE/feature_request.md

---
name: Feature request
about: Suggest an idea for this project
labels: Change Request

---

### Is your feature request related to a problem? Please describe.

A clear and concise description of what the problem is. Ex. I'm always frustrated when [...]

### Describe the solution you'd like

A clear and concise description of what you want to happen.

### Describe alternatives you've considered

A clear and concise description of any alternative solutions or features you've considered.

### Additional context

Add any other context or screenshots about the feature request here.

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/ISSUE_TEMPLATE/how-to-use.md

---
name: How to Use PHPWord
about: Find out how to use PHPWord
labels: WontFix

---

***Please do not use the issue tracker to ask how to use PHPWord.***

Documentation is available on [Read the Docs](https://phpword.readthedocs.io/en/latest/).

Sample code is in the [`/samples/` directory](https://github.com/PHPOffice/PHPWord/tree/develop/samples).

Usage questions belong on [Stack Overflow](https://stackoverflow.com/questions/tagged/phpword).

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/PULL_REQUEST_TEMPLATE.md

### Description

Please include a summary of the change and which issue is fixed. Please also include relevant motivation and context.

Fixes # (issue)

### Checklist:

- [ ] I have run `composer run-script check --timeout=0` and no errors were reported
- [ ] The new code is covered by unit tests (check build/coverage for coverage report)
- [ ] I have updated the documentation to describe the changes

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/support.yml

# Label used to mark issues as support requests
supportLabel: Question
# Comment to post on issues marked as support requests. Add a link
# to a support page, or set to `false` to disable
supportComment: >
  This looks like a support question. Please ask your support questions on
  [StackOverflow](http://stackoverflow.com/questions/tagged/phpword),
  or [Gitter](https://gitter.im/PHPOffice/PHPWord).

  Thank you for your contributions.

# Whether to close issues marked as support requests
close: true
# Whether to lock issues marked as support requests
lock: false

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/workflows/ci.yml

name: CI
on:
  - push
  - pull_request
jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        php-version:
          - '7.2'
          - "7.3"
          - "7.4"
          - "8.0"

    name: PHP ${{ matrix.php-version }}

    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup PHP, with composer and extensions
        uses: shivammathur/setup-php@v2
        with:
          php-version: ${{ matrix.php-version }}
          extensions: ctype, dom, gd, iconv, fileinfo, libxml, mbstring, simplexml, xml, xmlreader, xmlwriter, zip, zlib
          coverage: none

      - name: Get composer cache directory
        id: composer-cache
        run: echo "::set-output name=dir::$(composer config cache-files-dir)"

      - name: Cache composer dependencies
        uses: actions/cache@v2
        with:
          path: ${{ steps.composer-cache.outputs.dir }}
          key: ${{ runner.os }}-composer-${{ hashFiles('**/composer.lock') }}
          restore-keys: ${{ runner.os }}-composer-

      - name: Update PHPUnit version for PHP 8
        id: composer-lock
        if: "startsWith(matrix.php-version, '8.')"
        run: |
          rm -f composer.lock
          echo "::set-output name=flags::--ignore-platform-reqs"
          composer remove phpunit/phpunit --dev --no-update --no-interaction
          composer require phpunit/phpunit ^8.0 --dev --no-update

      - name: Install dependencies
        run: composer update --no-progress --prefer-dist --optimize-autoloader ${{ steps.composer-lock.outputs.flags }}

      - name: Update code to make PHPUnit 8 compatible
        if: "startsWith(matrix.php-version, '8.')"
        run: |
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/function setUpBeforeClass()$/function setUpBeforeClass(): void/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/function tearDownAfterClass()$/function tearDownAfterClass(): void/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/function setUp()$/function setUp(): void/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/function tearDown()$/function tearDown(): void/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/->assertContains(/->assertStringContainsString(/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e 's/->assertNotContains(/->assertStringNotContainsString(/' {} \;
          find ./tests/ -name "*.php" -type f -exec sed -i -e "s/->assertInternalType('array', /->assertIsArray(/" {} \;
          sed -i "s/\$this->addWarning('The @expectedException,/\/\/\$this->addWarning('The @expectedException,/" ./vendor/phpunit/phpunit/src/Framework/TestCase.php
          sed -i "s/self::createWarning('The optional \$delta/\/\/self::createWarning('The optional \$delta/" ./vendor/phpunit/phpunit/src/Framework/Assert.php

      - name: Setup problem matchers for PHP
        run: echo "::add-matcher::${{ runner.tool_cache }}/php.json"

      - name: Setup problem matchers for PHPUnit
        run: echo "::add-matcher::${{ runner.tool_cache }}/phpunit.json"

      - name: Configure matchers
        uses: mheap/phpunit-matcher-action@v1

      - name: Test with PHPUnit
        run: ./vendor/bin/phpunit --teamcity --no-coverage -c ./

  php-cs-fixer:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup PHP, with composer and extensions
        uses: shivammathur/setup-php@v2
        with:
          php-version: 7.4
          extensions: ctype, dom, gd, iconv, fileinfo, libxml, mbstring, simplexml, xml, xmlreader, xmlwriter, zip, zlib
          coverage: none
          tools: cs2pr

      - name: Get composer cache directory
        id: composer-cache
        run: echo "::set-output name=dir::$(composer config cache-files-dir)"

      - name: Cache composer dependencies
        uses: actions/cache@v2
        with:
          path: ${{ steps.composer-cache.outputs.dir }}
          key: ${{ runner.os }}-composer-${{ hashFiles('**/composer.lock') }}
          restore-keys: ${{ runner.os }}-composer-

      - name: Install dependencies
        run: composer install --no-progress --prefer-dist --optimize-autoloader

      - name: Code style with PHP-CS-Fixer
        run: ./vendor/bin/php-cs-fixer fix --dry-run --show-progress=none --using-cache=no -v --format=checkstyle | cs2pr

  phpcs:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup PHP, with composer and extensions
        uses: shivammathur/setup-php@v2
        with:
          php-version: 7.4
          extensions: ctype, dom, gd, iconv, fileinfo, libxml, mbstring, simplexml, xml, xmlreader, xmlwriter, zip, zlib
          coverage: none
          tools: cs2pr

      - name: Get composer cache directory
        id: composer-cache
        run: echo "::set-output name=dir::$(composer config cache-files-dir)"

      - name: Cache composer dependencies
        uses: actions/cache@v2
        with:
          path: ${{ steps.composer-cache.outputs.dir }}
          key: ${{ runner.os }}-composer-${{ hashFiles('**/composer.lock') }}
          restore-keys: ${{ runner.os }}-composer-

      - name: Install dependencies
        run: composer install --no-progress --prefer-dist --optimize-autoloader

      - name: Code style with PHP_CodeSniffer
        run: ./vendor/bin/phpcs src/ tests/ --standard=PSR2 -n --ignore=src/PhpWord/Shared/PCLZip --report=checkstyle | cs2pr

  coverage:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup PHP, with composer and extensions
        uses: shivammathur/setup-php@v2
        with:
          php-version: 7.4
          extensions: ctype, dom, gd, iconv, fileinfo, libxml, mbstring, simplexml, xml, xmlreader, xmlwriter, zip, zlib
          coverage: xdebug

      - name: Get composer cache directory
        id: composer-cache
        run: echo "::set-output name=dir::$(composer config cache-files-dir)"

      - name: Cache composer dependencies
        uses: actions/cache@v2
        with:
          path: ${{ steps.composer-cache.outputs.dir }}
          key: ${{ runner.os }}-composer-${{ hashFiles('**/composer.lock') }}
          restore-keys: ${{ runner.os }}-composer-

      - name: Install dependencies
        run: composer install --no-progress --prefer-dist --optimize-autoloader

      - name: Coverage
        run: |
          ./vendor/bin/phpunit --coverage-clover coverage-clover.xml
          curl -LO https://scrutinizer-ci.com/ocular.phar
          php ocular.phar code-coverage:upload --format=php-clover coverage-clover.xml

## File: word/PHP_Word/vendor/phpoffice/phpword/.github/workflows/github-pages.yml

name: GithHub Pages
on:
  push:
    tags:
      - '*'

jobs:
  github-pages:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup PHP, with composer and extensions
        uses: shivammathur/setup-php@v2
        with:
          php-version: 7.4
          coverage: none # remove xdebug

      - name: Build API documentation
        run: |
          curl -LO https://github.com/phpDocumentor/phpDocumentor/releases/download/v3.0.0/phpDocumentor.phar
          php phpDocumentor.phar --directory src/ --target docs/api

      - name: Deploy to GithHub Pages
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./docs/api

## File: word/PHP_Word/vendor/phpoffice/phpword/.github_changelog_generator

user=PHPOffice
project=PHPWord

since-tag=0.18.1
future-release=0.18.2

issues=false
pulls=true

## File: word/PHP_Word/vendor/phpoffice/phpword/.php_cs.dist

<?php

$finder = PhpCsFixer\Finder::create()
    ->notName('pclzip.lib.php')
    ->notName('OLERead.php')
    ->in('samples')
    ->in('src')
    ->in('tests');

return PhpCsFixer\Config::create()
        ->setRiskyAllowed(true)
        ->setFinder($finder)
        ->setRules(array(
            'array_syntax' => array('syntax' => 'long'),
            'binary_operator_spaces' => array('align_double_arrow' => true),
            'blank_line_after_namespace' => true,
            'blank_line_after_opening_tag' => false,
            'blank_line_before_return' => true,
            'braces' => true,
            'cast_spaces' => true,
            'class_definition' => true,
            'class_keyword_remove' => false, // ::class keyword gives us beter support in IDE
            'combine_consecutive_unsets' => true,
            'concat_space' => array('spacing' => 'one'),
            'declare_equal_normalize' => true,
            'declare_strict_types' => false, // Too early to adopt strict types
            'dir_constant' => true,
            'elseif' => true,
            'encoding' => true,
            'ereg_to_preg' => true,
            'full_opening_tag' => true,
            'function_declaration' => true,
            'function_typehint_space' => true,
            'general_phpdoc_annotation_remove' => false, // No use for that
            'hash_to_slash_comment' => true,
            'header_comment' => false, // We don't use common header in all our files
            'heredoc_to_nowdoc' => false, // Not sure about this one
            'is_null' => false, // Risky
            'include' => true,
            'indentation_type' => true,
            'line_ending' => true,
            'linebreak_after_opening_tag' => true,
            'lowercase_cast' => true,
            'lowercase_constants' => true,
            'lowercase_keywords' => true,
            'mb_str_functions' => false, // No, too dangerous to change that
            'method_argument_space' => true,
            'method_separation' => true,
            'modernize_types_casting' => true,
            'native_function_casing' => true,
            'native_function_invocation'=> false, // This is risky and seems to be micro-optimization that make code uglier so not worth it, at least for now
            'new_with_braces' => true,
            'no_alias_functions' => true,
            'no_blank_lines_after_class_opening' => true,
            'no_blank_lines_after_phpdoc' => true,
            'no_blank_lines_before_namespace' => false, // we want 1 blank line before namespace
            'no_closing_tag' => true,
            'no_empty_comment' => true,
            'no_empty_phpdoc' => true,
            'no_empty_statement' => true,
            'no_extra_consecutive_blank_lines' => array('break', 'continue', 'extra', 'return', 'throw', 'use', 'useTrait', 'curly_brace_block', 'parenthesis_brace_block', 'square_brace_block'),
            'no_leading_import_slash' => true,
            'no_leading_namespace_whitespace' => true,
            'no_mixed_echo_print' => true,
            'no_multiline_whitespace_around_double_arrow' => true,
            'no_multiline_whitespace_before_semicolons' => true,
            'no_php4_constructor' => true,
            'no_short_bool_cast' => true,
            'no_short_echo_tag' => true,
            'no_singleline_whitespace_before_semicolons' => true,
            'no_spaces_after_function_name' => true,
            'no_spaces_around_offset' => true,
            'no_spaces_inside_parenthesis' => true,
            'no_trailing_comma_in_list_call' => true,
            'no_trailing_comma_in_singleline_array' => true,
            'no_trailing_whitespace' => true,
            'no_trailing_whitespace_in_comment' => true,
            'no_unneeded_control_parentheses' => true,
            'no_unreachable_default_argument_value' => true,
            'no_unused_imports' => true,
            'no_useless_else' => true,
            'no_useless_return' => true,
            'no_whitespace_before_comma_in_array' => true,
            'no_whitespace_in_blank_line' => true,
            'normalize_index_brace' => true,
            'not_operator_with_space' => false, // No we prefer to keep '!' without spaces
            'not_operator_with_successor_space' => false, // idem
            'object_operator_without_whitespace' => true,
            'ordered_class_elements' => false, // We prefer to keep some freedom
            'ordered_imports' => true,
            'php_unit_construct' => true,
            'php_unit_dedicate_assert' => true,
            'php_unit_fqcn_annotation' => true,
            'php_unit_strict' => false, // We sometime actually need assertEquals
            'phpdoc_add_missing_param_annotation' => true,
            'phpdoc_align' => false, // Waste of time
            'phpdoc_annotation_without_dot' => true,
            'phpdoc_indent' => true,
            'phpdoc_inline_tag' => true,
            'phpdoc_no_access' => true,
            'phpdoc_no_alias_tag' => true,
            'phpdoc_no_empty_return' => true,
            'phpdoc_no_package' => true,
            'phpdoc_no_useless_inheritdoc' => true,
            'phpdoc_order' => true,
            'phpdoc_return_self_reference' => true,
            'phpdoc_scalar' => true,
            'phpdoc_separation' => false,
            'phpdoc_single_line_var_spacing' => true,
            'phpdoc_summary' => false,
            'phpdoc_to_comment' => true,
            'phpdoc_trim' => true,
            'phpdoc_types' => true,
            'phpdoc_var_without_name' => true,
            'pow_to_exponentiation' => false,
            'pre_increment' => false,
            'protected_to_private' => true,
            'psr0' => true,
            'psr4' => true,
            'random_api_migration' => false, // This breaks our unit tests
            'return_type_declaration' => true,
            'self_accessor' => true,
            'semicolon_after_instruction' => false, // Buggy in `samples/index.php`
            'short_scalar_cast' => true,
            'silenced_deprecation_error' => true,
            'simplified_null_return' => false, // While technically correct we prefer to be explicit when returning null
            'single_blank_line_at_eof' => true,
            'single_blank_line_before_namespace' => true,
            'single_class_element_per_statement' => true,
            'single_import_per_statement' => true,
            'single_line_after_imports' => true,
            'single_quote' => true,
            'space_after_semicolon' => true,
            'standardize_not_equals' => true,
            'strict_comparison' => false, // No, too dangerous to change that
            'strict_param' =>  false, // No, too dangerous to change that
            'switch_case_semicolon_to_colon' => true,
            'switch_case_space' => true,
            'ternary_operator_spaces' => true,
            'ternary_to_null_coalescing' => false, // Cannot use that with PHP 5.6
            'trailing_comma_in_multiline_array' => true,
            'trim_array_spaces' => false,
            'unary_operator_spaces' => true,
            'visibility_required' => true,
            'whitespace_after_comma_in_array' => true,
    ));

## File: word/PHP_Word/vendor/phpoffice/phpword/bootstrap.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors. test bootstrap
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

$vendorDirPath = realpath(__DIR__ . '/vendor');
if (file_exists($vendorDirPath . '/autoload.php')) {
    require $vendorDirPath . '/autoload.php';
} else {
    throw new Exception(
        sprintf(
            'Could not find file \'%s\'. It is generated by Composer. Use \'install --prefer-source\' or \'update --prefer-source\' Composer commands to move forward.',
            $vendorDirPath . '/autoload.php'
        )
    );
}

## File: word/PHP_Word/vendor/phpoffice/phpword/CHANGELOG.md

# Changelog
All notable changes to this project will be documented in this file.
This project adheres to [Semantic Versioning](http://semver.org/).

## [0.18.2](https://github.com/PHPOffice/PHPWord/tree/0.18.2) (2021-06-04)

[Full Changelog](https://github.com/PHPOffice/PHPWord/compare/0.18.1...0.18.2)

### Bug fixes
- when adding image to relationship first check that the generated RID is actually unique [\#2063](https://github.com/PHPOffice/PHPWord/pull/2063) ([tpv-ebben](https://github.com/tpv-ebben))
- Update chart, don't write 'c:overlap' if grouping is 'clustered' [\#2052](https://github.com/PHPOffice/PHPWord/pull/2052) ([dfsd534](https://github.com/dfsd534))
- Update Html parser to accept line-height:normal [\#2041](https://github.com/PHPOffice/PHPWord/pull/2041) ([joelgo](https://github.com/joelgo))
- Fix image border in Word2007 Writer for LibreOffice 7 [\#2021](https://github.com/PHPOffice/PHPWord/pull/2021) ([kamilmmach](https://github.com/kamilmmach))

### Miscellaneous
- Corrected namespace for Language class in docs. [\#2087](https://github.com/PHPOffice/PHPWord/pull/2087) ([MegaChriz](https://github.com/MegaChriz))
- Added support for Garamond font [\#2078](https://github.com/PHPOffice/PHPWord/pull/2078) ([artemkolotilkin](https://github.com/artemkolotilkin))
- Add BorderStyle for Cell Style to documentation [\#2090](https://github.com/PHPOffice/PHPWord/pull/2090) ([DShkrabak](https://github.com/DShkrabak))

## [0.18.1](https://github.com/PHPOffice/PHPWord/tree/0.18.1) (2021-03-08)

[Full Changelog](https://github.com/PHPOffice/PHPWord/compare/0.18.0...0.18.1)

### Bug fixes
- Fix BC break in #1946. This package does not replace laminas/laminas-zendframework-bridge [\#2032](https://github.com/PHPOffice/PHPWord/pull/2032) ([mussbach](https://github.com/mussbach))

## [0.18.0](https://github.com/PHPOffice/PHPWord/tree/0.18.0) (2021-02-12)

[Full Changelog](https://github.com/PHPOffice/PHPWord/compare/0.17.0...0.18.0)

### Enhancements
- Add support for charts in template processor [\#2012](https://github.com/PHPOffice/PHPWord/pull/2012) ([dbarzin](https://github.com/dbarzin))
- add/setting page element border style. [\#1986](https://github.com/PHPOffice/PHPWord/pull/1986) ([emnabs](https://github.com/emnabs))
- allow to use customized pdf library [\#1983](https://github.com/PHPOffice/PHPWord/pull/1983) ([SailorMax](https://github.com/SailorMax))
- feat: Update addHtml to handle style inheritance [\#1965](https://github.com/PHPOffice/PHPWord/pull/1965) ([Julien1138](https://github.com/Julien1138))
- Add parsing of Shape node values [\#1924](https://github.com/PHPOffice/PHPWord/pull/1924) ([sven-ahrens](https://github.com/sven-ahrens))
- Allow to redefine TCPDF object [\#1907](https://github.com/PHPOffice/PHPWord/pull/1907) ([SailorMax](https://github.com/SailorMax))
- Enhancements to addHTML parser [\#1902](https://github.com/PHPOffice/PHPWord/pull/1902) ([lubosdz](https://github.com/lubosdz))
- Make Default Paper Configurable [\#1851](https://github.com/PHPOffice/PHPWord/pull/1851) ([oleibman](https://github.com/oleibman))
- Implement various missing features for the ODT writer [\#1796](https://github.com/PHPOffice/PHPWord/pull/1796) ([oleibman](https://github.com/oleibman))
- Added support for "cloudConvert" images [\#1794](https://github.com/PHPOffice/PHPWord/pull/1794) ([ErnestStaug](https://github.com/ErnestStaug))
- Add support for several features for the RTF writer [\#1775](https://github.com/PHPOffice/PHPWord/pull/1775) ([oleibman](https://github.com/oleibman))
- Add font style for Field elements [\#1774](https://github.com/PHPOffice/PHPWord/pull/1774) ([oleibman](https://github.com/oleibman))
- Add support for ListItemRun in HTML writer [\#1766](https://github.com/PHPOffice/PHPWord/pull/1766) ([stefan-91](https://github.com/stefan-91))
- Improvements in RTF writer [\#1755](https://github.com/PHPOffice/PHPWord/pull/1755) ([oleibman](https://github.com/oleibman))
- Allow a closure to be passed with image replacement tags [\#1716](https://github.com/PHPOffice/PHPWord/pull/1716) ([mbardelmeijer](https://github.com/mbardelmeijer))
- Add Option for Dynamic Chart Legend Position [\#1699](https://github.com/PHPOffice/PHPWord/pull/1699) ([Stephan212](https://github.com/Stephan212))
- Add parsing of HTML checkbox input field [\#1832](https://github.com/PHPOffice/PHPWord/pull/1832) ([Matze2010](https://github.com/Matze2010))

### Bug fixes
- Fix image stroke in libreoffice 7.x [\#1992](https://github.com/PHPOffice/PHPWord/pull/1992) ([Adizbek](https://github.com/Adizbek))
- Fix deprecated warning for non-hexadecimal number [\#1988](https://github.com/PHPOffice/PHPWord/pull/1988) ([Ciki](https://github.com/Ciki))
- Fix limit not taken into account when adding image in template [\#1967](https://github.com/PHPOffice/PHPWord/pull/1967) ([jsochor](https://github.com/jsochor))
- Add null check when setComplexValue is not found [\#1936](https://github.com/PHPOffice/PHPWord/pull/1936) ([YannikFirre](https://github.com/YannikFirre))
- Some document have non-standard locale code [\#1824](https://github.com/PHPOffice/PHPWord/pull/1824) ([ErnestStaug](https://github.com/ErnestStaug))
- Fixes PHPDoc @param and @return types for several Converter methods [\#1818](https://github.com/PHPOffice/PHPWord/pull/1818) ([caugner](https://github.com/caugner))
- Update the regexp to avoid catastrophic backtracking [\#1809](https://github.com/PHPOffice/PHPWord/pull/1809) ([juzser](https://github.com/juzser))
- Fix PHPUnit tests on develop branch [\#1771](https://github.com/PHPOffice/PHPWord/pull/1771) ([mdupont](https://github.com/mdupont))
- TemplateProcessor cloneBlock wrongly clones images [\#1763](https://github.com/PHPOffice/PHPWord/pull/1763) ([alarai](https://github.com/alarai))

### Miscellaneous
- Compatibility with PHP 7.4, PHP 8.0 and migrate to Laminas Escaper [\#1946](https://github.com/PHPOffice/PHPWord/pull/1946) ([liborm85](https://github.com/liborm85))
- Remove legacy PHPOffice/Common package, fix PHP 8.0 compatibility [\#1996](https://github.com/PHPOffice/PHPWord/pull/1996) ([liborm85](https://github.com/liborm85))
- Improve Word2007 Test Coverage [\#1858](https://github.com/PHPOffice/PHPWord/pull/1858) ([oleibman](https://github.com/oleibman))
- Fix typo in docs. Update templates-processing.rst [\#1952](https://github.com/PHPOffice/PHPWord/pull/1952) ([mnvx](https://github.com/mnvx))
- Fix documentation and method name for FootnoteProperties [\#1776](https://github.com/PHPOffice/PHPWord/pull/1776) ([mdupont](https://github.com/mdupont))
- fix: documentation about paragraph indentation [\#1764](https://github.com/PHPOffice/PHPWord/pull/1764) ([mdupont](https://github.com/mdupont))
- Update templates-processing.rst [\#1745](https://github.com/PHPOffice/PHPWord/pull/1745) ([igronus](https://github.com/igronus))
- Unused variables $rows, $cols in sample [\#1877](https://github.com/PHPOffice/PHPWord/pull/1877) ([ThanasisMpalatsoukas](https://github.com/ThanasisMpalatsoukas))
- Add unit test for NumberingStyle [\#1744](https://github.com/PHPOffice/PHPWord/pull/1744) ([Manunchik](https://github.com/Manunchik))
- Add unit test for PhpWord Settings [\#1743](https://github.com/PHPOffice/PHPWord/pull/1743) ([Manunchik](https://github.com/Manunchik))
- Add unit test for Media elements [\#1742](https://github.com/PHPOffice/PHPWord/pull/1742) ([Manunchik](https://github.com/Manunchik))
- Update templates processing docs [\#1729](https://github.com/PHPOffice/PHPWord/pull/1729) ([hcdias](https://github.com/hcdias))

v0.17.0 (01 oct 2019)
----------------------
### Added
- Add methods setValuesFromArray and cloneRowFromArray to the TemplateProcessor @geraldb-nicat #670
- Set complex type in template @troosan #1565
- implement support for section vAlign @troosan #1569
- ParseStyle for border-color @Gllrm0 #1551
- Html writer auto invert text color @SailorMax #1387
- Add RightToLeft table presentation. @troosan #1550
- Add support for page vertical alignment. @troosan #672 #1569
- Adding setNumId method for ListItem style @eweso #1329
- Add support for basic fields in RTF writer. @Samuel-BF #1717

### Fixed
- Fix HTML border-color parsing. @troosan #1551 #1570
- Language::validateLocale should pass with locale 'zxx'. @efpapado #1558
- can't align center vertically with the text @ter987 #672
- fix parsing of border-color and add test @troosan #1570
- TrackChange doesn't handle all return types of \DateTime::createFromFormat(...) @superhaggis #1584
- To support PreserveText inside sub container @bhattnishant #1637
- No nested w:pPr elements in ListItemRun. @waltertamboer #1628
- Ensure that entity_loader disable variable is re-set back to the original setting @seamuslee001 #1585

### Miscellaneous
- Use embedded http server to test loading of remote images @troosan #1544
- Change private to protected to be able extending class Html @SpinyMan #1646
- Fix apt-get crash in Travis CI for PHP 5.3 @mdupont #1707

v0.16.0 (30 dec 2018)
----------------------
### Added
- Add getVariableCount method in TemplateProcessor. @nicoder #1272
- Add setting Chart Title and Legend visibility @Tom-Magill #1433
- Add ability to pass a Style object in Section constructor @ndench #1416
- Add support for hidden text @Alexmg86 #1527
- Add support for setting images in TemplateProcessor @SailorMax #1170
- Add "Plain Text" type to SDT (Structured Document Tags) @morrisdj #1541
- Added possibility to index variables inside cloned block in TemplateProcessor @JPBetley #817
- Added possibility to replace variables inside cloned block with values in TemplateProcessor @DIDoS #1392

### Fixed
- Fix regex in `cloneBlock` function @nicoder #1269
- HTML Title Writer loses text when Title contains a TextRun instead a string. @begnini #1436
- Fix regex in fixBrokenMacros, make it less greedy @MuriloSo @brainwood @yurii-sio2 #1502 #1345
- 240 twips are being added to line spacing, should not happen when using lineRule fixed @troosan #1509 #1505
- Adding table layout to the generated HTML @aarangara #1441
- Fix loading of Sharepoint document @Garrcomm #1498
- RTF writer: Round getPageSizeW and getPageSizeH to avoid decimals @Patrick64 #1493
- Fix parsing of Office 365 documents @Timanx #1485
- For RTF writers, sizes should should never have decimals @Samuel-BF #1536
- Style Name Parsing fails if document generated by a non-english word version @begnini #1434

### Miscellaneous
- Get rid of duplicated code in TemplateProcessor @abcdmitry #1161

v0.15.0 (14 Jul 2018)
----------------------
### Added
- Parsing of `align` HTML attribute - @troosan #1231
- Parse formatting inside HTML lists - @troosan @samimussbach #1239 #945 #1215 #508
- Parsing of CSS `direction` instruction, HTML `lang` attribute, formatting inside table cell - @troosan #1273 #1252 #1254
- Add support for Track changes @Cip @troosan #354 #1262
- Add support for fixed Table Layout @aoloe @ekopach @troosan #841 #1276
- Add support for Cell Spacing @dox07 @troosan #1040
- Add parsing of formatting inside lists @atomicalnet @troosan #594
- Added support for Vertically Raised or Lowered Text (w:position) @anrikun @troosan #640
- Add support for MACROBUTTON field @phryneas @troosan #1021
- Add support for Hyphenation @Trainmaster #1282 (Document: `autoHyphenation`, `consecutiveHyphenLimit`, `hyphenationZone`, `doNotHyphenateCaps`, Paragraph: `suppressAutoHyphens`)
- Added support for Floating Table Positioning (tblpPr) @anrikun #639
- Added support for Image text wrapping distance @troosan #1310
- Added parsing of CSS line-height and text-indent in HTML reader @troosan #1316
- Added the ability to enable gridlines and axislabels on charts @FrankMeyer #576
- Add support for table indent (tblInd) @Trainmaster #1343
- Added parsing of internal links in HTML reader @lalop #1336
- Several improvements to charts @JAEK-S #1332
- Add parsing of html image in base64 format @jgpATs2w #1382
- Added Support for Indentation & Tabs on RTF Writer. @smaug1985 #1405
- Allows decimal numbers in HTML line-height style @jgpATs2w #1413

### Fixed
- Fix reading of docx default style - @troosan #1238
- Fix the size unit of when parsing html images - @troosan #1254
- Fixed HTML parsing of nested lists - @troosan #1265
- Save PNG alpha information when using remote images. @samsullivan #779
- Fix parsing of `<w:br/>` tag. @troosan #1274
- Bookmark are not writton as internal link in html writer @troosan #1263
- It should be possible to add a Footnote in a ListItemRun @troosan #1287 #1287
- Fix colspan and rowspan for tables in HTML Writer @mattbolt #1292
- Fix parsing of Heading and Title formating @troosan @gthomas2 #465
- Fix Dateformat typo, fix hours casing, add Month-Day-Year formats @ComputerTinker #591
- Support reading of w:drawing for documents produced by word 2011+ @gthomas2 #464 #1324
- Fix missing column width in ODText writer @potofcoffee #413
- Disable entity loader before parsing XML to avoid XXE injection @Tom4t0 #1427

### Changed
- Remove zend-stdlib dependency @Trainmaster #1284
- The default unit for `\PhpOffice\PhpWord\Style\Image` changed from `px` to `pt`.

### Miscellaneous
- Drop GitHub pages, switch to coveralls for code coverage analysis @czosel #1360

v0.14.0 (29 Dec 2017)
----------------------
This release fixes several bugs and adds some new features.
This version brings compatibility with PHP 7.0 & 7.1

### Added
- Possibility to control the footnote numbering - @troosan #1068
- Image creation from string - @troosan #937
- Introduced the `\PhpOffice\PhpWord\SimpleType\NumberFormat` simple type. - @troosan
- Support for ContextualSpacing - @postHawk #1088
- Possiblity to hide spelling and/or grammatical errors - @troosan #542
- Possiblity to set default document language as well as changing the language for each text element - @troosan #1108
- Support for Comments - @troosan #1067
- Support for paragraph textAlignment - @troosan #1165
- Add support for HTML underline tag <u> in addHtml - @zNightFalLz #1186
- Add support for HTML <br> in addHtml - @anrikun @troosan #659
- Allow to change cell width unit - guillaume-ro-fr #986
- Allow to change the line height rule @troosan
- Implement PageBreak for odt writer @cookiekiller #863 #824
- Allow to force an update of all fields on opening a document - @troosan #951
- Allow adding a CheckBox in a TextRun - @irond #727
- Add support for HTML img tag - @srggroup #934
- Add support for password protection for docx - @mariahaubner #1019

### Fixed
- Loosen dependency to Zend
- Images are not being printed when generating PDF - @hubertinio #1074 #431
- Fixed some PHP 7 warnings - @	likeuntomurphy #927
- Fixed PHP 7.2 compatibility (renamed `Object` class names to `ObjectElement`) - @SailorMax #1185
- Fixed Word 97 reader - @alsofronie @Benpxpx @mario-rivera #912 #920 #892
- Fixed image loading over https - @troosan #988
- Impossibility to set different even and odd page headers - @troosan #981
- Fixed Word2007 reader where unnecessary paragraphs were being created - @donghaobo #1043 #620
- Fixed Word2007 reader where margins were not being read correctly - @slowprog #885 #1008
- Impossible to add element PreserveText in Section - @rvanlaak #452
- Added missing options for numbering format - @troosan #1041
- Fixed impossibility to set a different footer for first page - @ctrlaltca #1116, @aoloe #875
- Fixed styles not being applied by HTML writer, better pdf output - @sarke #1047 #500 #1139
- Fixed read docx error when document contains image from remote url - @FBnil #1173 #1176
- Padded the $args array to remove error - @kaigoh #1150, @reformed #870
- Fix incorrect image size between windows and mac - @bskrtich #874
- Fix adding HTML table to document - @mogilvie @arivanbastos #324
- Fix parsing on/off values (w:val="true|false|1|0|on|off") - @troosan #1221 #1219
- Fix error on Empty Dropdown Entry - @ComputerTinker #592

### Deprecated
- PhpWord->getProtection(), get it from the settings instead PhpWord->getSettings()->getDocumentProtection();

v0.13.0 (31 July 2016)
-------------------
This release brings several improvements in `TemplateProcessor`, automatic output escaping feature for OOXML, ODF, HTML, and RTF (turned off, by default).
It also introduces constants for horizontal alignment options, and resolves some issues with PHP 7.
Manual installation feature has been dropped since the release. Please, use [Composer](https://getcomposer.org/) to install PHPWord.

### Added
- Introduced the `\PhpOffice\PhpWord\SimpleType\Jc` simple type. - @RomanSyroeshko
- Introduced the `\PhpOffice\PhpWord\SimpleType\JcTable` simple type. - @RomanSyroeshko
- Introduced writer for the "Paragraph Alignment" element (see `\PhpOffice\PhpWord\Writer\Word2007\Element\ParagraphAlignment`). - @RomanSyroeshko
- Introduced writer for the "Table Alignment" element (see `\PhpOffice\PhpWord\Writer\Word2007\Element\TableAlignment`). - @RomanSyroeshko
- Supported indexed arrays in arguments of `TemplateProcessor::setValue()`. - @RomanSyroeshko #618
- Introduced automatic output escaping for OOXML, ODF, HTML, and RTF. To turn the feature on use `phpword.ini` or `\PhpOffice\PhpWord\Settings`. - @RomanSyroeshko #483
- Supported processing of headers and footers in `TemplateProcessor::applyXslStyleSheet()`. - @RomanSyroeshko #335

### Changed
- Improved error message for the case when `autoload.php` is not found. - @RomanSyroeshko #371
- Renamed the `align` option of `NumberingLevel`, `Frame`, `Table`, and `Paragraph` styles into `alignment`. - @RomanSyroeshko
- Improved performance of `TemplateProcessor::setValue()`. - @kazitanvirahsan #614, #617
- Fixed some HTML tags not rendering any output (p, header & table) - #257, #324 - @twmobius and @garethellis

### Deprecated
- `getAlign` and `setAlign` methods of `NumberingLevel`, `Frame`, `Table`, and `Paragraph` styles.
Use the correspondent `getAlignment` and `setAlignment` methods instead. - @RomanSyroeshko
- `left`, `right`, and `justify` alignment options for paragraphs (now are mapped to `Jc::START`, `Jc::END`, and `Jc::BOTH`). - @RomanSyroeshko
- `left`, `right`, and `justify` alignment options for tables (now are mapped to `Jc::START`, `Jc::END`, and `Jc::CENTER`). - @RomanSyroeshko
- `TCPDF` due to its limited HTML support. Use `DomPDF` or `MPDF` writer instead. - @RomanSyroeshko #399

### Removed
- `\PhpOffice\PhpWord\Style\Alignment`. Style properties, which previously stored instances of this class, now deal with strings.
In each case set of available string values is defined by the correspondent simple type. - @RomanSyroeshko
- Manual installation support. Since the release we have dependencies on third party libraries,
so installation via ZIP-archive download is not an option anymore. To install PHPWord use [Composer](https://getcomposer.org/).
 We also removed `\PhpOffice\PhpWord\Autoloader`, because the latter change made it completely useless.
 Autoloaders provided by Composer are in use now (see `bootstrap.php`). - @RomanSyroeshko
- `\PhpOffice\PhpWord\Shared\Drawing` replaced by `\PhpOffice\Common\Drawing`. - @Progi1984 #658
- `\PhpOffice\PhpWord\Shared\Font`. - @Progi1984 #658
- `\PhpOffice\PhpWord\Shared\String` replaced by `\PhpOffice\Common\Text`. - @Progi1984 @RomanSyroeshko #658
- `\PhpOffice\PhpWord\Shared\XMLReader` replaced by `\PhpOffice\Common\XMLReader`. - @Progi1984 #658
- `\PhpOffice\PhpWord\Shared\XMLWriter` replaced by `\PhpOffice\Common\XMLWriter`. - @Progi1984 @RomanSyroeshko #658
- `AbstractContainer::addMemoryImage()`. Use `AbstractContainer::addImage()` instead.

### Fixed
- `Undefined property` error while reading MS-DOC documents. - @jaberu #610
- Corrupted OOXML template issue in case when its names is broken immediately after `$` sign.
That case wasn't taken into account in implementation of `TemplateProcessor::fixBrokenMacros()`. - @RomanSyroeshko @d-damien #548

v0.12.1 (30 August 2015)
-----------------------
Maintenance release. This release is focused primarily on `TemplateProcessor`.

### Changes
- Changed visibility of all private properties and methods of `TemplateProcessor` to `protected`. - @RomanSyroeshko #498
- Improved performance of `TemplateProcessor::setValue()`. - @RomanSyroeshko @nicoSWD #513

### Bugfixes
- Fixed issue with "Access denied" message while opening `Sample_07_TemplateCloneRow.docx` and `Sample_23_TemplateBlock.docx` result files on Windows platform. - @RomanSyroeshko @AshSat #532
- Fixed `PreserveText` element alignment in footer (see `Sample_12_HeaderFooter.php`). - @RomanSyroeshko @SSchwaiger #495

v0.12.0 (3 January 2015)
-----------------------
This release added form fields (textinput, checkbox, and dropdown), drawing shapes (arc, curve, line, polyline, rect, oval), and basic 2D chart (pie, doughnut, bar, line, area, scatter, radar) elements along with some new styles. Basic MsDoc reader is introduced.

### Features
- Element: Ability to add drawing shapes (arc, curve, line, polyline, rect, oval) using new `Shape` element - @ivanlanin #123
- Font: New `scale`, `spacing`, and `kerning` property of font style - @ivanlanin
- Paragraph:  Added shading to the paragraph style for full width shading - @lrobert #264
- RTF Writer: Support for sections, margins, and borders - @ivanlanin #249
- Section: Ability to set paper size, e.g. A4, A3, and Legal - @ivanlanin #249
- General: New `PhpWord::save()` method to encapsulate `IOFactory` - @ivanlanin
- General: New `Shared\Converter` static class - @ivanlanin
- Chart: Basic 2D chart (pie, doughnut, bar, line, area, scatter, radar) - @ivanlanin #278
- Chart: 3D charts and ability to set width and height - @ivanlanin
- FormField: Ability to add textinput, checkbox, and dropdown form elements - @ivanlanin #266
- Setting: Ability to define document protection (readOnly, comments, trackedChanges, forms) - @ivanlanin
- Setting: Ability to remove [Compatibility Mode] text in the MS Word title bar - @ivanlanin
- SDT: Ability to add structured document tag elements (comboBox, dropDownList, date) - @ivanlanin
- Paragraph: Support for paragraph with borders - @ivanlanin #294
- Word2007 Writer : Support for RTL - @Progi1984 #331
- MsDOC Reader: Basic MsDOC Reader - @Progi1984 #23, #287
- "absolute" horizontal and vertical positioning of Frame - @basjan #302
- Add new-page function for PDF generation. For multiple PDF-backends - @chc88 #426
- Report style options enumerated when style unknown - @h6w

### Bugfixes
- Fix rare PclZip/realpath/PHP version problem - @andrew-kzoo #261
- `addHTML` encoding and ampersand fixes for PHP 5.3 - @bskrtich #270
- Page breaks on titles and tables - @ivanlanin #274
- Table inside vertical border does not rendered properly - @ivanlanin #280
- `add<elementName>` of container should be case insensitive, e.g. `addToc` should be accepted, not only `addTOC` - @ivanlanin #294
- Fix specific borders (and margins) were not written correctly in word2007 writer - @pscheit #327
- "HTML is not a valid writer" exception while running "Sample_36_RTL.php" - @RomanSyroeshko #340
- "addShape()" magic method in AbstractContainer is mistakenly named as "addObject()" - @GMTA #356
- `Element\Section::setPageSizeW()` and `Element\Section::setPageSizeH()` were mentioned in the docs but not implemented.
- Special Characters (ampersand) in Title break docx output - @RomanSyroeshko #401
- `<th>` tag is closed with `</td>` tag: - @franzholz #438

### Deprecated
- `Element\Link::getTarget()` replaced by `Element\Link::getSource()`
- `Element\Section::getSettings()` and `Element\Section::setSettings()` replaced by `Element\Section::getStyle()` and `Element\Section::setStyle()`
- `Shared\Drawing` and `Shared\Font` merged into `Shared\Converter`
- `DocumentProperties` replaced by `Metadata\DocInfo`
- `Template` replaced by `TemplateProcessor`
- `PhpWord->loadTemplate($filename)`

### Miscellaneous
- Docs: Add known issue on `README` about requirement for temporary folder to be writable and update `samples/index.php` for this requirement check - @ivanlanin #238
- Docs: Correct elements.rst about Line - @chrissharkman #292
- PclZip: Remove temporary file after used - @andrew-kzoo #265
- Autoloader: Add the ability to set the autoloader options - @bskrtich #267
- Element: Refactor elements to move set relation Id from container to element - @ivanlanin
- Introduced CreateTemporaryFileException, CopyFileException - @RomanSyroeshko
- Settings: added method to set user defined temporary directory - @RomanSyroeshko #310
- Renamed `Template` into `TemplateProcessor` - @RomanSyroeshko #216
- Reverted #51. All text escaping must be performed out of the library - @RomanSyroeshko #51

v0.11.1 (2 June 2014)
--------------------
This is an immediate bugfix release for HTML reader.

- HTML Reader: `<p>` and header tags puts no output - @canyildiz @ivanlanin #257

v0.11.0 (1 June 2014)
--------------------
This release marked the change of PHPWord license from LGPL 2.1 to LGPL 3. Four new elements were added: TextBox, ListItemRun, Field, and Line. Relative and absolute positioning for images and textboxes were added. Writer classes were refactored into parts, elements, and styles. ODT and RTF features were enhanced. Ability to add elements to PHPWord object via HTML were implemented. RTF and HTML reader were initiated.

### Features
- Image: Ability to define relative and absolute positioning - @basjan #217
- Footer: Conform footer with header by adding firstPage, evenPage and by inheritance - @basjan @ivanlanin #219
- Element: New `TextBox` element - @basjan @ivanlanin #228, #229, #231
- HTML: Ability to add elements to PHPWord object via html - @basjan #231
- Element: New `ListItemRun` element that can add a list item with inline formatting like a textrun - @basjan #235
- Table: Ability to add table inside a cell (nested table) - @ivanlanin #149
- RTF Writer: UTF8 support for RTF: Internal UTF8 text is converted to Unicode before writing - @ivanlanin #158
- Table: Ability to define table width (in percent and twip) and position - @ivanlanin #237
- RTF Writer: Ability to add links and page breaks in RTF - @ivanlanin #196
- ListItemRun: Remove fontStyle parameter because ListItemRun is inherited from TextRun and TextRun doesn't have fontStyle - @ivanlanin
- Config: Ability to use a config file to store various common settings - @ivanlanin #200
- ODT Writer: Enable inline font style in TextRun - @ivanlanin
- ODT Writer: Enable underline, strike/doublestrike, smallcaps/allcaps, superscript/subscript font style - @ivanlanin
- ODT Writer: Enable section and column - @ivanlanin
- PDF Writer: Add TCPDF and mPDF as optional PDF renderer library - @ivanlanin
- ODT Writer: Enable title element and custom document properties - @ivanlanin
- ODT Reader: Ability to read standard and custom document properties - @ivanlanin
- Word2007 Writer: Enable the missing custom document properties writer - @ivanlanin
- Image: Enable "image float left" - @ivanlanin #244
- RTF Writer: Ability to write document properties - @ivanlanin
- RTF Writer: Ability to write image - @ivanlanin
- Element: New `Field` element - @basjan #251
- RTF Reader: Basic RTF reader - @ivanlanin #72, #252
- Element: New `Line` element - @basjan #253
- Title: Ability to apply numbering in heading - @ivanlanin #193
- HTML Reader: Basic HTML reader - @ivanlanin #80, #254
- RTF Writer: Basic table writing - @ivanlanin #245

### Bugfixes
- Header: All images added to the second header were assigned to the first header - @basjan #222
- Conversion: Fix conversion from cm to pixel, pixel to cm, and pixel to point - @basjan #233, #234
- PageBreak: Page break adds new line in the beginning of the new page - @ivanlanin #150
- Image: `marginLeft` and `marginTop` cannot accept float value - @ivanlanin #248
- Title: Orphan `w:fldChar` caused OpenOffice to crash when opening DOCX - @ivanlanin #236

### Deprecated
- Static classes `Footnotes`, `Endnotes`, and `TOC`
- `Writer\Word2007\Part`: `Numbering::writeNumbering()`, `Settings::writeSettings()`, `WebSettings::writeWebSettings()`, `ContentTypes::writeContentTypes()`, `Styles::writeStyles()`, `Document::writeDocument()` all changed into `write()`
- `Writer\Word2007\Part\DocProps`: Split into `Writer\Word2007\Part\DocPropsCore` and `Writer\Word2007\Part\DocPropsApp`
- `Element\Title::getBookmarkId()` replaced by `Element\Title::getRelationId()`
- `Writer\HTML::writeDocument`: Replaced by `Writer\HTML::getContent`

### Miscellaneous
- License: Change the project license from LGPL 2.1 into LGPL 3.0 - #211
- Word2007 Writer: New `Style\Image` class - @ivanlanin
- Refactor: Replace static classes `Footnotes`, `Endnotes`, and `TOC` with `Collections` - @ivanlanin #206
- QA: Reactivate `phpcpd` and `phpmd` on Travis - @ivanlanin
- Refactor: PHPMD recommendation: Change all `get...` method that returns `boolean` into `is...` or `has...` - @ivanlanin
- Docs: Create gh-pages branch for API documentation - @Progi1984 #154
- QA: Add `.scrutinizer.yml` and include `composer.lock` for preparation to Scrutinizer - @ivanlanin #186
- Writer: Refactor writer parts using composite pattern - @ivanlanin
- Docs: Show code quality and test code coverage badge on README
- Style: Change behaviour of `set...` function of boolean properties; when none is defined, assumed true - @ivanlanin
- Shared: Unify PHP ZipArchive and PCLZip features into PhpWord ZipArchive - @ivanlanin
- Docs: Create VERSION file - @ivanlanin
- QA: Improve dan update requirement check in `samples` folder - @ivanlanin

v0.10.1 (21 May 2014)
--------------------
This is a bugfix release for `php-zip` requirement in Composer.

- Change Composer requirements for php-zip from `require` to `suggest` - @bskrtich #246

v0.10.0 (4 May 2014)
-------------------
This release marked heavy refactorings on internal code structure with the creation of some abstract classes to reduce code duplication. `Element` subnamespace is introduced in this release to replace `Section`. Word2007 reader capability is greatly enhanced. Endnote is introduced. List numbering is now customizable. Basic HTML and PDF writing support is enabled. Basic ODText reader is introduced.

### Features
- Image: Get image dimensions without EXIF extension - @andrew-kzoo #184
- Table: Add `tblGrid` element for Libre/Open Office table sizing - @gianis6 #183
- Footnote: Ability to insert textbreak in footnote `$footnote->addTextBreak()` - @ivanlanin
- Footnote: Ability to style footnote reference mark by using `FootnoteReference` style - @ivanlanin
- Font: Add `bgColor` to font style to define background using HEX color - @jcarignan #168
- Table: Add `exactHeight` to row style to define whether row height should be exact or atLeast - @jcarignan #168
- Element: New `CheckBox` element for sections and table cells - @ozilion #156
- Settings: Ability to use PCLZip as alternative to ZipArchive - @bskrtich @ivanlanin #106, #140, #185
- Template: Ability to find & replace variables in headers & footers - @dgudgeon #190
- Template: Ability to clone & delete block of text using `cloneBlock` and `deleteBlock` - @diego-vieira #191
- TOC: Ability to have two or more TOC in one document and to set min and max depth for TOC - @Pyreweb #189
- Table: Ability to add footnote in table cell - @ivanlanin #187
- Footnote: Ability to add image in footnote - @ivanlanin #187
- ListItem: Ability to add list item in header/footer - @ivanlanin #187
- CheckBox: Ability to add checkbox in header/footer - @ivanlanin #187
- Link: Ability to add link in header/footer - @ivanlanin #187
- Object: Ability to add object in header, footer, textrun, and footnote - @ivanlanin #187
- Media: Add `Media::resetElements()` to reset all media data - @juzi #19
- General: Add `Style::resetStyles()` - @ivanlanin #187
- DOCX Reader: Ability to read header, footer, footnotes, link, preservetext, textbreak, pagebreak, table, list, image, and title - @ivanlanin
- Endnote: Ability to add endnotes - @ivanlanin
- ListItem: Ability to create custom list and reset list number - @ivanlanin #10, #198
- ODT Writer: Basic table writing support - @ivanlanin
- Image: Keep image aspect ratio if only 1 dimension styled - @japonicus #194
- HTML Writer: Basic HTML writer: text, textrun, link, title, textbreak, table, image (as Base64), footnote, endnote - @ivanlanin #203, #67, #147
- PDF Writer: Basic PDF writer using DomPDF: All HTML element except image - @ivanlanin #68
- DOCX Writer: Change `docProps/app.xml` `Application` to `PHPWord` - @ivanlanin
- DOCX Writer: Create `word/settings.xml` and `word/webSettings.xml` dynamically - @ivanlanin
- ODT Writer: Basic image writing - @ivanlanin
- ODT Writer: Link writing - @ivanlanin
- ODT Reader: Basic ODText Reader - @ivanlanin #71
- Section: Ability to define gutter and line numbering - @ivanlanin
- Font: Small caps, all caps, and double strikethrough - @ivanlanin #151
- Settings: Ability to use measurement unit other than twips with `setMeasurementUnit` - @ivanlanin #199
- Style: Remove `bgColor` from `Font`, `Table`, and `Cell` and put it into the new `Shading` style - @ivanlanin
- Style: New `Indentation` and `Spacing` style - @ivanlanin
- Paragraph: Ability to define first line and right indentation - @ivanlanin

### Bugfixes
- Footnote: Footnote content doesn't show footnote reference number - @ivanlanin #170
- Documentation: Error in a function - @theBeerNut #195

### Deprecated
- `createTextRun` replaced by `addTextRun`
- `createFootnote` replaced by `addFootnote`
- `createHeader` replaced by `addHeader`
- `createFooter` replaced by `addFooter`
- `createSection` replaced by `addSection`
- `Element\Footnote::getReferenceId` replaced by `Element\AbstractElement::getRelationId`
- `Element\Footnote::setReferenceId` replaced by `Element\AbstractElement::setRelationId`
- `Footnote::addFootnoteLinkElement` replaced by `Media::addElement`
- `Footnote::getFootnoteLinkElements` replaced by `Media::getElements`
- All current methods on `Media`
- `Element\Link::getLinkSrc` replaced by `Element\Link::getTarget`
- `Element\Link::getLinkName` replaced by `Element\Link::getText`
- `Style\Cell::getDefaultBorderColor`

### Miscellaneous
- Documentation: Simplify page level docblock - @ivanlanin #179
- Writer: Refactor writer classes and create a new `Write\AbstractWriter` abstract class - @ivanlanin #160
- General: Refactor folders: `Element` and `Exception` - @ivanlanin #187
- General: Remove legacy `HashTable` and `Shared\ZipStreamWrapper` and all related properties/methods - @ivanlanin #187
- Element: New `AbstractElement` abstract class - @ivanlanin #187
- Media: Refactor media class to use one method for all docPart (section, header, footer, footnote) - @ivanlanin #187
- General: Remove underscore prefix from all private properties name - @ivanlanin #187
- General: Move Section `Settings` to `Style\Section` - @ivanlanin #187
- General: Give `Abstract` prefix and `Interface` suffix for all abstract classes and interfaces as per [PHP-FIG recommendation](https://github.com/php-fig/fig-standards/blob/master/bylaws/002-psr-naming-conventions.md) - @ivanlanin #187
- Style: New `Style\AbstractStyle` abstract class - @ivanlanin #187
- Writer: New 'ODText\Base` class - @ivanlanin #187
- General: Rename `Footnote` to `Footnotes` to reflect the nature of collection - @ivanlanin
- General: Add some unit tests for Shared & Element (100%!) - @Progi1984
- Test: Add some samples and tests for image wrapping style - @brunocasado #59
- Refactor: Remove Style\Tabs - @ivanlanin
- Refactor: Apply composite pattern for writers - @ivanlanin
- Refactor: Split `AbstractContainer` from `AbstractElement` - @ivanlanin
- Refactor: Apply composite pattern for Word2007 reader - @ivanlanin

v0.9.1 (27 Mar 2014)
-------------------
This is a bugfix release for PSR-4 compatibility.

- Fixed PSR-4 composer autoloader - @AntonTyutin

v0.9.0 (26 Mar 2014)
-------------------
This release marked the transformation to namespaces (PHP 5.3+).

### Features
- Image: Ability to use remote or GD images using `addImage()` on sections, headers, footer, cells, and textruns - @ivanlanin
- Header: Ability to use remote or GD images using `addWatermark()` - @ivanlanin

### Bugfixes
- Preserve text doesn't render correctly when the text is not the first word, e.g. 'Page {PAGE}' - @ivanlanin

### Miscellaneous
- Move documentation to [Read The Docs](http://phpword.readthedocs.org/en/develop/) - @Progi1984 @ivanlanin #82
- Reorganize and redesign samples folder - @ivanlanin #137
- Use `PhpOffice\PhpWord` namespace for PSR compliance - @RomanSyroeshko @gabrielbull #159, #58
- Restructure folders and change folder name `Classes` to `src` and `Tests` to `test` for PSR compliance - @RomanSyroeshko @gabrielbull
- Compliance to phpDocumentor - @ivanlanin
- Merge Style\TableFull into Style\Table. Style\TableFull is deprecated - @ivanlanin #160
- Merge Section\MemoryImage into Section\Image. Section\Image is deprecated - @ivanlanin #160

v0.8.1 (17 Mar 2014)
-------------------
This is a bugfix release for image detection functionality.

- Added fallback for computers that do not have exif_imagetype - @bskrtich, @gabrielbull

v0.8.0 (15 Mar 2014)
-------------------
This release merged a lot of improvements from the community. Unit tests introduced in this release and has reached 90% code coverage.

### Features
- Template: Permit to save a template generated as a file (PHPWord_Template::saveAs()) - @RomanSyroeshko #56, #57
- Word2007: Support sections page numbering - @gabrielbull
- Word2007: Added line height methods to mirror the line height settings in Word in the paragraph styling - @gabrielbull
- Word2007: Added support for page header & page footer height - @JillElaine #5
- General: Add ability to manage line breaks after image insertion - @bskrtich #6, #66, #84
- Template: Ability to limit number of replacements performed by setValue() method of Template class - @RomanSyroeshko #52, #53, #85
- Table row: Repeat as header row & allow row to break across pages - @ivanlanin #48, #86
- Table: Table width in percentage - @ivanlanin #48, #86
- Font: Superscript and subscript - @ivanlanin #48, #86
- Paragraph: Hanging paragraph - @ivanlanin #48, #86
- Section: Multicolumn and section break - @ivanlanin #48, #86
- Template: Ability to apply XSL style sheet to Template - @RomanSyroeshko #46, #47, #83
- General: PHPWord_Shared_Font::pointSizeToTwips() converter - @ivanlanin #87
- Paragraph: Ability to define normal paragraph style with PHPWord::setNormalStyle() - @ivanlanin #87
- Paragraph: Ability to define parent style (basedOn) and style for following paragraph (next) - @ivanlanin #87
- Clone table rows on the fly when using a template document - @jeroenmoors #44, #88
- Initial addition of basic footnote support - @deds #16
- Paragraph: Ability to define paragraph pagination: widow control, keep next, keep lines, and page break before - @ivanlanin #92
- General: PHPWord_Style_Font refactoring - @ivanlanin #93
- Font: Use points instead of halfpoints internally. Conversion to halfpoints done during XML Writing. - @ivanlanin #93
- Paragraph: setTabs() function - @ivanlanin #92
- General: Basic support for TextRun on ODT and RTF - @ivanlanin #99
- Reader: Basic Reader for Word2007 - @ivanlanin #104
- TextRun: Allow Text Break in Text Run - @bskrtich  #109
- General: Support for East Asian fontstyle - @jhfangying #111, #118
- Image: Use exif_imagetype to check image format instead of extension name - @gabrielbull #114
- General: Setting for XMLWriter Compatibility option - @bskrtich  #103
- MemoryImage: Allow remote image when allow_url_open = on - @ivanlanin #122
- TextBreak: Allow font and paragraph style for text break - @ivanlanin #18

### Bugfixes
- Fixed bug with cell styling - @gabrielbull
- Fixed bug list items inside of cells - @gabrielbull
- Adding a value that contains "&" in a template breaks it - @SiebelsTim #51
- Example in README.md is broken - @Progi1984 #89
- General: PHPWord_Shared_Drawing::centimetersToPixels() conversion - @ivanlanin #94
- Footnote: Corrupt DOCX reported by MS Word when sections > 1 and not every sections have footnote - @ivanlanin #125

### Miscellaneous
- UnitTests - @Progi1984

v0.7.0 (28 Jan 2014)
-------------------
This is the first release after a long development hiatus in [CodePlex](https://phpword.codeplex.com/). This release initialized ODT and RTF Writer, along with some other new features for the existing Word2007 Writer, e.g. tab, multiple header, rowspan and colspan. [Composer](https://packagist.org/packages/phpoffice/phpword) and [Travis](https://travis-ci.org/PHPOffice/PHPWord) were added.

### Features
- Implement RTF Writer - @Progi1984 #1
- Implement ODT Writer - @Progi1984 #2
- Word2007: Add rowspan and colspan to cells - @kaystrobach
- Word2007: Support for tab stops - @RLovelett
- Word2007: Support Multiple headers - @RLovelett
- Word2007: Wrapping Styles to Images - @gabrielbull
- Added support for image wrapping style - @gabrielbull

### Bugfixes
- "Warning: Invalid error type specified in ...\PHPWord.php on line 226" is thrown when the specified template file is not found - @RomanSyroeshko #32
- PHPWord_Shared_String.IsUTF8 returns FALSE for Cyrillic UTF-8 input - @RomanSyroeshko #34
- Temporary files naming logic in PHPWord_Template can lead to a collision - @RomanSyroeshko #38

### Miscellaneous
- Add superscript/subscript styling in Excel2007 Writer - @MarkBaker
- add indentation support to paragraphs - @deds
- Support for Composer - @Progi1984 #27
- Basic CI with Travis - @Progi1984
- Added PHPWord_Exception and exception when could not copy the template - @Progi1984
- IMPROVED: Moved examples out of Classes directory - @Progi1984
- IMPROVED: Advanced string replace in setValue for Template - @Esmeraldo [#49](http://phpword.codeplex.com/workitem/49)

## File: word/PHP_Word/vendor/phpoffice/phpword/composer.json

{
    "name": "phpoffice/phpword",
    "description": "PHPWord - A pure PHP library for reading and writing word processing documents (OOXML, ODF, RTF, HTML, PDF)",
    "keywords": [
        "PHP", "PHPOffice", "office", "PHPWord", "word", "template", "template processor", "reader", "writer",
        "docx", "OOXML", "OpenXML", "Office Open XML", "ISO IEC 29500", "WordprocessingML",
        "RTF", "Rich Text Format", "doc", "odt", "ODF", "OpenDocument", "PDF", "HTML"
    ],
    "homepage": "http://phpoffice.github.io",
    "type": "library",
    "license": "LGPL-3.0",
    "authors": [
        {
            "name": "Mark Baker"
        },
        {
            "name": "Gabriel Bull",
            "email": "me@gabrielbull.com",
            "homepage": "http://gabrielbull.com/"
        },
        {
            "name": "Franck Lefevre",
            "homepage": "https://rootslabs.net/blog/"
        },
        {
            "name": "Ivan Lanin",
            "homepage": "http://ivan.lanin.org"
        },
        {
            "name": "Roman Syroeshko",
            "homepage": "http://ru.linkedin.com/pub/roman-syroeshko/34/a53/994/"
        },
        {
            "name": "Antoine de Troostembergh"
        }
    ],
    "scripts": {
        "test": [
            "phpunit --color=always"
        ],
        "test-no-coverage": [
            "phpunit --color=always --no-coverage"
        ],
        "check": [
            "php-cs-fixer fix --ansi --dry-run --diff",
            "phpcs --report-width=200 --report-summary --report-full samples/ src/ tests/ --ignore=src/PhpWord/Shared/PCLZip --standard=PSR2 -n",
            "phpmd src/,tests/ text ./phpmd.xml.dist --exclude pclzip.lib.php",
            "@test-no-coverage"
        ],
        "fix": [
            "php-cs-fixer fix --ansi"
        ]
    },
    "scripts-descriptions": {
        "test": "Runs all unit tests",
        "test-no-coverage": "Runs all unit tests, without code coverage",
        "check": "Runs PHP CheckStyle and PHP Mess detector",
        "fix": "Fixes issues found by PHP-CS"
    },
    "require": {
        "php": "^5.3.3 || ^7.0 || ^8.0",
        "ext-xml": "*",
        "laminas/laminas-escaper": "^2.2"
    },
    "require-dev": {
        "ext-zip": "*",
        "ext-gd": "*",
        "phpunit/phpunit": "^4.8.36 || ^7.0",
        "squizlabs/php_codesniffer": "^2.9 || ^3.5",
        "friendsofphp/php-cs-fixer": "^2.2",
        "phpmd/phpmd": "2.*",
        "phploc/phploc": "2.* || 3.* || 4.* || 5.* || 6.* || 7.*",
        "dompdf/dompdf":"0.8.* || 1.0.*",
        "tecnickcom/tcpdf": "6.*",
        "mpdf/mpdf": "5.7.4 || 6.* || 7.* || 8.*",
        "php-coveralls/php-coveralls": "1.1.0 || ^2.0"
    },
    "suggest": {
        "ext-zip": "Allows writing OOXML and ODF",
        "ext-gd2": "Allows adding images",
        "ext-xmlwriter": "Allows writing OOXML and ODF",
        "ext-xsl": "Allows applying XSL style sheet to headers, to main document part, and to footers of an OOXML template",
        "dompdf/dompdf": "Allows writing PDF"
    },
    "autoload": {
        "psr-4": {
            "PhpOffice\\PhpWord\\": "src/PhpWord"
        }
    },
    "extra": {
        "branch-alias": {
            "dev-develop": "0.19-dev"
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/CONTRIBUTING.md

# Contributing to PHPWord

PHPWord is built by the crowd and for the crowd. Every contribution is welcome; either by [reporting a bug](https://github.com/PHPOffice/PHPWord/issues/new?labels=Bug+Report&template=bug_report.md) or [suggesting improvements](https://github.com/PHPOffice/PHPWord/issues/new?labels=Change+Request&template=feature_request.md), or in a more active form like [requesting a pull](https://github.com/PHPOffice/PHPWord/pulls).

We want to create a high quality document writer and reader library that people can use with more confidence and fewer bugs. We want to collaborate happily, code joyfully, and live merrily. Thus, below are some guidelines that we expect to be followed by each contributor:

- **Be brief, but be bold**. State your issues briefly. But speak out your ideas loudly, even if you can't or don't know how to implement them right away. The world will be better with limitless innovations.
- **Follow PHP-FIG standards**. We follow PHP Standards Recommendations (PSRs) by [PHP Framework Interoperability Group](http://www.php-fig.org/). If you're not familiar with these standards, [familiarize yourself now](https://github.com/php-fig/fig-standards). Also, please run `composer fix` to automatically fix your code to match these recommendations.
- **Test your code**. No one knows your code better than you, so we depend on you to test the changes you make before pull request submission. We use [PHPUnit](https://phpunit.de/) for our testing purposes and request that you use this tool too. Tests can be ran with `composer test`. [Documentation for writing tests with PHPUnit is available on Read the Docs.](https://phpunit.readthedocs.io)
- **Use best practices when submitting pull requests**. Create a separate branch named specifically for the issue that you are addressing. Read the [GitHub manual](https://help.github.com/articles/about-pull-requests) to learn more about pull requests and GitHub. If you are new to GitHub, read [this short manual](https://help.github.com/articles/fork-a-repo) to get yourself familiar with forks and how git works in general. [This video](http://www.youtube.com/watch?v=-zvHQXnBO6c) explains how to synchronize your fork on GitHub with the upstream branch from PHPWord.

## Getting Started

1. [Clone](https://help.github.com/en/articles/cloning-a-repository) [PHPWord](https://github.com/PHPOffice/PHPWord/)
2. [Install Composer](https://getcomposer.org/download/) if you don't already have it
3. Open your terminal and:
  1. Switch to the directory PHPWord was cloned to (e.g., `cd ~/Projects/PHPWord/`)
  2. Run `composer install` to install the dependencies

You're ready to start working on PHPWord! Tests belong in the `/tests/PhpWord/` directory, the source code is in `/src/PhpWord/`, and any documentation should go in `/docs/`. Familiarize yourself with the codebase and try your hand at fixing [one of our outstanding issues](https://github.com/PHPOffice/PHPWord/issues). Before you get started, check the [existing pull requests](https://github.com/PHPOffice/PHPWord/pulls) to make sure no one else is already working on it.

Once you have an issue you want to start working on, you'll need to write tests for it, and then you can start implementing the changes necessary to pass the new tests. To run the tests, you can run one of the following commands in your terminal:

- `composer test-no-coverage` to run all of the tests
- `composer test` to run all of the tests and generate test coverage reports

When you're ready to submit your new (and fully tested) feature, ensure `composer check` passes and [submit a pull request to PHPWord](https://github.com/PHPOffice/PHPWord/issues/new).

That's it. Thank you for your interest in PHPWord, and welcome!

May the Force be with you.

## File: word/PHP_Word/vendor/phpoffice/phpword/COPYING

                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.

## File: word/PHP_Word/vendor/phpoffice/phpword/COPYING.LESSER

                   GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.

  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

       0) Convey the Minimal Corresponding Source under the terms of this
       License, and the Corresponding Application Code in a form
       suitable for, and under terms that permit, the user to
       recombine or relink the Application with a modified version of
       the Linked Version to produce a modified Combined Work, in the
       manner specified by section 6 of the GNU GPL for conveying
       Corresponding Source.

       1) Use a suitable shared library mechanism for linking with the
       Library.  A suitable mechanism is one that (a) uses at run time
       a copy of the Library already present on the user's computer
       system, and (b) will operate properly with a modified version
       of the Library that is interface-compatible with the Linked
       Version.

   e) Provide Installation Information, but only if you would otherwise
   be required to provide such information under section 6 of the
   GNU GPL, and only to the extent that such information is
   necessary to install and execute a modified version of the
   Combined Work produced by recombining or relinking the
   Application with a modified version of the Linked Version. (If
   you use option 4d0, the Installation Information must accompany
   the Minimal Corresponding Source and Corresponding Application
   Code. If you use option 4d1, you must provide the Installation
   Information in the manner specified by section 6 of the GNU GPL
   for conveying Corresponding Source.)

  5. Combined Libraries.

  You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

   a) Accompany the combined library with a copy of the same work based
   on the Library, uncombined with any other library facilities,
   conveyed under the terms of this License.

   b) Give prominent notice with the combined library that part of it
   is a work based on the Library, and explaining where to find the
   accompanying uncombined form of the same work.

  6. Revised Versions of the GNU Lesser General Public License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number. If the
Library as you received it specifies that a certain numbered version
of the GNU Lesser General Public License "or any later version"
applies to it, you have the option of following the terms and
conditions either of that published version or of any later version
published by the Free Software Foundation. If the Library as you
received it does not specify a version number of the GNU Lesser
General Public License, you may choose any version of the GNU Lesser
General Public License ever published by the Free Software Foundation.

  If the Library as you received it specifies that a proxy can decide
whether future versions of the GNU Lesser General Public License shall
apply, that proxy's public statement of acceptance of any version is
permanent authorization for you to choose that version for the
Library.

## File: word/PHP_Word/vendor/phpoffice/phpword/LICENSE

PHPWord, a pure PHP library for reading and writing word processing documents.

Copyright (c) 2010-2016 PHPWord.

PHPWord is free software: you can redistribute it and/or modify
it under the terms of the GNU Lesser General Public License version 3 as published by
the Free Software Foundation.

PHPWord is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Lesser General Public License version 3 for more details.

You should have received a copy of the GNU Lesser General Public License version 3
along with PHPWord. If not, see <http://www.gnu.org/licenses/>.

## File: word/PHP_Word/vendor/phpoffice/phpword/phpword.ini.dist

; Default config file for PHPWord
; Copy this file into phpword.ini and use Settings::loadConfig to load

[General]

compatibility         = true
zipClass              = ZipArchive
pdfRendererName       = DomPDF
pdfRendererPath       =
; tempDir               = "C:\PhpWordTemp"
outputEscapingEnabled = false

[Font]

defaultFontName = Arial
defaultFontSize = 10

[Paper]

defaultPaper = "A4"

## File: word/PHP_Word/vendor/phpoffice/phpword/README.md

# ![PHPWord](https://rawgit.com/PHPOffice/PHPWord/develop/docs/images/phpword.svg "PHPWord")

Master:
[![Latest Stable Version](https://poser.pugx.org/phpoffice/phpword/v/stable.png)](https://packagist.org/packages/phpoffice/phpword)
[![Build Status](https://travis-ci.org/PHPOffice/PHPWord.svg?branch=master)](https://travis-ci.org/PHPOffice/PHPWord)
[![Code Quality](https://scrutinizer-ci.com/g/PHPOffice/PHPWord/badges/quality-score.png?b=master)](https://scrutinizer-ci.com/g/PHPOffice/PHPWord/)
[![Coverage Status](https://coveralls.io/repos/github/PHPOffice/PHPWord/badge.svg?branch=master)](https://coveralls.io/github/PHPOffice/PHPWord?branch=master)
[![Total Downloads](https://poser.pugx.org/phpoffice/phpword/downloads.png)](https://packagist.org/packages/phpoffice/phpword)
[![License](https://poser.pugx.org/phpoffice/phpword/license.png)](https://packagist.org/packages/phpoffice/phpword)
[![Join the chat at https://gitter.im/PHPOffice/PHPWord](https://img.shields.io/badge/GITTER-join%20chat-green.svg)](https://gitter.im/PHPOffice/PHPWord)

Develop:
[![Latest Development Version](https://img.shields.io/badge/unstable-dev--develop-orange.svg)](https://packagist.org/packages/phpoffice/phpword#dev-develop)
[![Build Status](https://travis-ci.org/PHPOffice/PHPWord.svg?branch=develop)](https://travis-ci.org/PHPOffice/PHPWord/branches)
[![Code Quality](https://scrutinizer-ci.com/g/PHPOffice/PHPWord/badges/quality-score.png?b=develop)](https://scrutinizer-ci.com/g/PHPOffice/PHPWord/?branch=develop)
[![Coverage Status](https://coveralls.io/repos/github/PHPOffice/PHPWord/badge.svg?branch=develop)](https://coveralls.io/github/PHPOffice/PHPWord?branch=develop)

PHPWord is a library written in pure PHP that provides a set of classes to write to and read from different document file formats. The current version of PHPWord supports Microsoft [Office Open XML](http://en.wikipedia.org/wiki/Office_Open_XML) (OOXML or OpenXML), OASIS [Open Document Format for Office Applications](http://en.wikipedia.org/wiki/OpenDocument) (OpenDocument or ODF), [Rich Text Format](http://en.wikipedia.org/wiki/Rich_Text_Format) (RTF), HTML, and PDF.

PHPWord is an open source project licensed under the terms of [LGPL version 3](COPYING.LESSER). PHPWord is aimed to be a high quality software product by incorporating [continuous integration](https://travis-ci.org/PHPOffice/PHPWord) and [unit testing](http://phpoffice.github.io/PHPWord/coverage/develop/). You can learn more about PHPWord by reading the [Developers' Documentation](http://phpword.readthedocs.org/).

If you have any questions, please ask on [StackOverFlow](https://stackoverflow.com/questions/tagged/phpword)

Read more about PHPWord:

- [Features](#features)
- [Requirements](#requirements)
- [Installation](#installation)
- [Getting started](#getting-started)
- [Contributing](#contributing)
- [Developers' Documentation](http://phpword.readthedocs.org/)

## Features

With PHPWord, you can create OOXML, ODF, or RTF documents dynamically using your PHP 5.3.3+ scripts. Below are some of the things that you can do with PHPWord library:

- Set document properties, e.g. title, subject, and creator.
- Create document sections with different settings, e.g. portrait/landscape, page size, and page numbering
- Create header and footer for each sections
- Set default font type, font size, and paragraph style
- Use UTF-8 and East Asia fonts/characters
- Define custom font styles (e.g. bold, italic, color) and paragraph styles (e.g. centered, multicolumns, spacing) either as named style or inline in text
- Insert paragraphs, either as a simple text or complex one (a text run) that contains other elements
- Insert titles (headers) and table of contents
- Insert text breaks and page breaks
- Insert and format images, either local, remote, or as page watermarks
- Insert binary OLE Objects such as Excel or Visio
- Insert and format table with customized properties for each rows (e.g. repeat as header row) and cells (e.g. background color, rowspan, colspan)
- Insert list items as bulleted, numbered, or multilevel
- Insert hyperlinks
- Insert footnotes and endnotes
- Insert drawing shapes (arc, curve, line, polyline, rect, oval)
- Insert charts (pie, doughnut, bar, line, area, scatter, radar)
- Insert form fields (textinput, checkbox, and dropdown)
- Create document from templates
- Use XSL 1.0 style sheets to transform headers, main document part, and footers of an OOXML template
- ... and many more features on progress

## Requirements

PHPWord requires the following:

- PHP 5.3.3+
- [XML Parser extension](http://www.php.net/manual/en/xml.installation.php)
- [Laminas Escaper component](https://docs.laminas.dev/laminas-escaper/intro/)
- [Zip extension](http://php.net/manual/en/book.zip.php) (optional, used to write OOXML and ODF)
- [GD extension](http://php.net/manual/en/book.image.php) (optional, used to add images)
- [XMLWriter extension](http://php.net/manual/en/book.xmlwriter.php) (optional, used to write OOXML and ODF)
- [XSL extension](http://php.net/manual/en/book.xsl.php) (optional, used to apply XSL style sheet to template )
- [dompdf library](https://github.com/dompdf/dompdf) (optional, used to write PDF)

## Installation

PHPWord is installed via [Composer](https://getcomposer.org/).
To [add a dependency](https://getcomposer.org/doc/04-schema.md#package-links) to PHPWord in your project, either

Run the following to use the latest stable version
```sh
    composer require phpoffice/phpword
```
or if you want the latest master version
```sh
    composer require phpoffice/phpword:dev-master
```

You can of course also manually edit your composer.json file
```json
{
    "require": {
       "phpoffice/phpword": "v0.18.*"
    }
}
```

## Getting started

The following is a basic usage example of the PHPWord library.

```php
<?php
require_once 'bootstrap.php';

// Creating the new document...
$phpWord = new \PhpOffice\PhpWord\PhpWord();

/* Note: any element you append to a document must reside inside of a Section. */

// Adding an empty Section to the document...
$section = $phpWord->addSection();
// Adding Text element to the Section having font styled by default...
$section->addText(
    '"Learn from yesterday, live for today, hope for tomorrow. '
        . 'The important thing is not to stop questioning." '
        . '(Albert Einstein)'
);

/*
 * Note: it's possible to customize font style of the Text element you add in three ways:
 * - inline;
 * - using named font style (new font style object will be implicitly created);
 * - using explicitly created font style object.
 */

// Adding Text element with font customized inline...
$section->addText(
    '"Great achievement is usually born of great sacrifice, '
        . 'and is never the result of selfishness." '
        . '(Napoleon Hill)',
    array('name' => 'Tahoma', 'size' => 10)
);

// Adding Text element with font customized using named font style...
$fontStyleName = 'oneUserDefinedStyle';
$phpWord->addFontStyle(
    $fontStyleName,
    array('name' => 'Tahoma', 'size' => 10, 'color' => '1B2232', 'bold' => true)
);
$section->addText(
    '"The greatest accomplishment is not in never falling, '
        . 'but in rising again after you fall." '
        . '(Vince Lombardi)',
    $fontStyleName
);

// Adding Text element with font customized using explicitly created font style object...
$fontStyle = new \PhpOffice\PhpWord\Style\Font();
$fontStyle->setBold(true);
$fontStyle->setName('Tahoma');
$fontStyle->setSize(13);
$myTextElement = $section->addText('"Believe you can and you\'re halfway there." (Theodor Roosevelt)');
$myTextElement->setFontStyle($fontStyle);

// Saving the document as OOXML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'Word2007');
$objWriter->save('helloWorld.docx');

// Saving the document as ODF file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'ODText');
$objWriter->save('helloWorld.odt');

// Saving the document as HTML file...
$objWriter = \PhpOffice\PhpWord\IOFactory::createWriter($phpWord, 'HTML');
$objWriter->save('helloWorld.html');

/* Note: we skip RTF, because it's not XML-based and requires a different example. */
/* Note: we skip PDF, because "HTML-to-PDF" approach is used to create PDF documents. */
```

More examples are provided in the [samples folder](samples/). For an easy access to those samples launch `php -S localhost:8000` in the samples directory then browse to [http://localhost:8000](http://localhost:8000) to view the samples.
You can also read the [Developers' Documentation](http://phpword.readthedocs.org/) for more detail.

## Contributing

We welcome everyone to contribute to PHPWord. Below are some of the things that you can do to contribute.

- Read [our contributing guide](CONTRIBUTING.md).
- [Fork us](https://github.com/PHPOffice/PHPWord/fork) and [request a pull](https://github.com/PHPOffice/PHPWord/pulls) to the [develop](https://github.com/PHPOffice/PHPWord/tree/develop) branch.
- Submit [bug reports or feature requests](https://github.com/PHPOffice/PHPWord/issues) to GitHub.
- Follow [@PHPWord](https://twitter.com/PHPWord) and [@PHPOffice](https://twitter.com/PHPOffice) on Twitter.

## File: word/PHP_Word/vendor/phpoffice/phpword/sonar-project.properties

# must be unique in a given SonarQube instance
sonar.projectKey=phpoffice:phpword
# this is the name and version displayed in the SonarQube UI. Was mandatory prior to SonarQube 6.1.
sonar.projectName=PHPWord
sonar.projectVersion=0.16

# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
# This property is optional if sonar.modules is set.
sonar.sources=src
sonar.tests=tests
sonar.php.coverage.reportPaths=build/logs/clover.xml
sonar.php.tests.reportPath=build/logs/logfile.xml

# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8

sonar.host.url=http://localhost:9000

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/AbstractCollection.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Collection abstract class
 *
 * @since 0.10.0
 */
abstract class AbstractCollection
{
    /**
     * Items
     *
     * @var \PhpOffice\PhpWord\Element\AbstractContainer[]
     */
    private $items = array();

    /**
     * Get items
     *
     * @return \PhpOffice\PhpWord\Element\AbstractContainer[]
     */
    public function getItems()
    {
        return $this->items;
    }

    /**
     * Get item by index
     *
     * @param int $index
     * @return \PhpOffice\PhpWord\Element\AbstractContainer
     */
    public function getItem($index)
    {
        if (array_key_exists($index, $this->items)) {
            return $this->items[$index];
        }

        return null;
    }

    /**
     * Set item.
     *
     * @param int $index
     * @param \PhpOffice\PhpWord\Element\AbstractContainer $item
     */
    public function setItem($index, $item)
    {
        if (array_key_exists($index, $this->items)) {
            $this->items[$index] = $item;
        }
    }

    /**
     * Add new item
     *
     * @param \PhpOffice\PhpWord\Element\AbstractContainer $item
     * @return int
     */
    public function addItem($item)
    {
        $index = $this->countItems() + 1;
        $this->items[$index] = $item;

        return $index;
    }

    /**
     * Get item count
     *
     * @return int
     */
    public function countItems()
    {
        return count($this->items);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Bookmarks.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Bookmarks collection
 *
 * @since 0.12.0
 */
class Bookmarks extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Charts.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Charts collection
 *
 * @since 0.12.0
 */
class Charts extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Comments.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Comments collection
 *
 * @since 0.12.0
 */
class Comments extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Endnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Endnotes collection
 *
 * @since 0.10.0
 */
class Endnotes extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Footnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Footnotes collection
 *
 * @since 0.10.0
 */
class Footnotes extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Collection/Titles.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Collection;

/**
 * Titles collection
 *
 * @since 0.10.0
 */
class Titles extends AbstractCollection
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/ComplexType/FootnoteProperties.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\ComplexType;

use PhpOffice\PhpWord\SimpleType\NumberFormat;

/**
 * Footnote properties
 *
 * @see http://www.datypic.com/sc/ooxml/e-w_footnotePr-1.html
 */
final class FootnoteProperties
{
    const RESTART_NUMBER_CONTINUOUS = 'continuous';
    const RESTART_NUMBER_EACH_SECTION = 'eachSect';
    const RESTART_NUMBER_EACH_PAGE = 'eachPage';

    const POSITION_PAGE_BOTTOM = 'pageBottom';
    const POSITION_BENEATH_TEXT = 'beneathText';
    const POSITION_SECTION_END = 'sectEnd';
    const POSITION_DOC_END = 'docEnd';

    /**
     * Footnote Positioning Location
     *
     * @var string
     */
    private $pos;

    /**
     * Footnote Numbering Format w:numFmt, one of PhpOffice\PhpWord\SimpleType\NumberFormat
     *
     * @var string
     */
    private $numFmt;

    /**
     * Footnote and Endnote Numbering Starting Value
     *
     * @var float
     */
    private $numStart;

    /**
     * Footnote and Endnote Numbering Restart Location
     *
     * @var string
     */
    private $numRestart;

    /**
     * Get the Footnote Positioning Location
     *
     * @return string
     */
    public function getPos()
    {
        return $this->pos;
    }

    /**
     * Set the Footnote Positioning Location (pageBottom, beneathText, sectEnd, docEnd)
     *
     * @param string $pos
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setPos($pos)
    {
        $position = array(
            self::POSITION_PAGE_BOTTOM,
            self::POSITION_BENEATH_TEXT,
            self::POSITION_SECTION_END,
            self::POSITION_DOC_END,
        );

        if (in_array($pos, $position)) {
            $this->pos = $pos;
        } else {
            throw new \InvalidArgumentException('Invalid value, on of ' . implode(', ', $position) . ' possible');
        }

        return $this;
    }

    /**
     * Get the Footnote Numbering Format
     *
     * @return string
     */
    public function getNumFmt()
    {
        return $this->numFmt;
    }

    /**
     * Set the Footnote Numbering Format
     *
     * @param string $numFmt One of NumberFormat
     * @return self
     */
    public function setNumFmt($numFmt)
    {
        NumberFormat::validate($numFmt);
        $this->numFmt = $numFmt;

        return $this;
    }

    /**
     * Get the Footnote Numbering Format
     *
     * @return float
     */
    public function getNumStart()
    {
        return $this->numStart;
    }

    /**
     * Set the Footnote Numbering Format
     *
     * @param float $numStart
     * @return self
     */
    public function setNumStart($numStart)
    {
        $this->numStart = $numStart;

        return $this;
    }

    /**
     * Get the Footnote and Endnote Numbering Starting Value
     *
     * @return string
     */
    public function getNumRestart()
    {
        return $this->numRestart;
    }

    /**
     * Set the Footnote and Endnote Numbering Starting Value (continuous, eachSect, eachPage)
     *
     * @param  string $numRestart
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setNumRestart($numRestart)
    {
        $restartNumbers = array(
            self::RESTART_NUMBER_CONTINUOUS,
            self::RESTART_NUMBER_EACH_SECTION,
            self::RESTART_NUMBER_EACH_PAGE,
        );

        if (in_array($numRestart, $restartNumbers)) {
            $this->numRestart = $numRestart;
        } else {
            throw new \InvalidArgumentException('Invalid value, on of ' . implode(', ', $restartNumbers) . ' possible');
        }

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/ComplexType/ProofState.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\ComplexType;

/**
 * Spelling and Grammatical Checking State
 *
 * @see http://www.datypic.com/sc/ooxml/e-w_proofState-1.html
 */
final class ProofState
{
    /**
     * Check Completed
     */
    const CLEAN = 'clean';

    /**
     * Check Not Completed
     */
    const DIRTY = 'dirty';

    /**
     * Spell Checking State
     *
     * @var string
     */
    private $spelling;

    /**
     * Grammatical Checking State
     *
     * @var string
     */
    private $grammar;

    /**
     * Set the Spell Checking State (dirty or clean)
     *
     * @param string $spelling
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setSpelling($spelling)
    {
        if ($spelling == self::CLEAN || $spelling == self::DIRTY) {
            $this->spelling = $spelling;
        } else {
            throw new \InvalidArgumentException('Invalid value, dirty or clean possible');
        }

        return $this;
    }

    /**
     * Get the Spell Checking State
     *
     * @return string
     */
    public function getSpelling()
    {
        return $this->spelling;
    }

    /**
     * Set the Grammatical Checking State (dirty or clean)
     *
     * @param string $grammar
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setGrammar($grammar)
    {
        if ($grammar == self::CLEAN || $grammar == self::DIRTY) {
            $this->grammar = $grammar;
        } else {
            throw new \InvalidArgumentException('Invalid value, dirty or clean possible');
        }

        return $this;
    }

    /**
     * Get the Grammatical Checking State
     *
     * @return string
     */
    public function getGrammar()
    {
        return $this->grammar;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/ComplexType/TblWidth.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\ComplexType;

use PhpOffice\PhpWord\SimpleType\TblWidth as TblWidthSimpleType;

/**
 * @see http://www.datypic.com/sc/ooxml/t-w_CT_TblWidth.html
 */
final class TblWidth
{
    /** @var string */
    private $type;

    /** @var int */
    private $value;

    /**
     * @param int $value If omitted, then its value shall be assumed to be 0
     * @param string $type If omitted, then its value shall be assumed to be dxa
     */
    public function __construct($value = 0, $type = TblWidthSimpleType::TWIP)
    {
        $this->value = $value;
        TblWidthSimpleType::validate($type);
        $this->type = $type;
    }

    /**
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * @return int
     */
    public function getValue()
    {
        return $this->value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/ComplexType/TrackChangesView.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\ComplexType;

/**
 * Visibility of Annotation Types
 *
 * @see http://www.datypic.com/sc/ooxml/e-w_revisionView-1.html
 */
final class TrackChangesView
{
    /**
     * Display Visual Indicator Of Markup Area
     *
     * @var bool
     */
    private $markup;

    /**
     * Display Comments
     *
     * @var bool
     */
    private $comments;

    /**
     * Display Content Revisions
     *
     * @var bool
     */
    private $insDel;

    /**
     * Display Formatting Revisions
     *
     * @var bool
     */
    private $formatting;

    /**
     * Display Ink Annotations
     *
     * @var bool
     */
    private $inkAnnotations;

    /**
     * Get Display Visual Indicator Of Markup Area
     *
     * @return bool True if markup is shown
     */
    public function hasMarkup()
    {
        return $this->markup;
    }

    /**
     * Set Display Visual Indicator Of Markup Area
     *
     * @param bool $markup
     *            Set to true to show markup
     */
    public function setMarkup($markup)
    {
        $this->markup = $markup === null ? true : $markup;
    }

    /**
     * Get Display Comments
     *
     * @return bool True if comments are shown
     */
    public function hasComments()
    {
        return $this->comments;
    }

    /**
     * Set Display Comments
     *
     * @param bool $comments
     *            Set to true to show comments
     */
    public function setComments($comments)
    {
        $this->comments = $comments === null ? true : $comments;
    }

    /**
     * Get Display Content Revisions
     *
     * @return bool True if content revisions are shown
     */
    public function hasInsDel()
    {
        return $this->insDel;
    }

    /**
     * Set Display Content Revisions
     *
     * @param bool $insDel
     *            Set to true to show content revisions
     */
    public function setInsDel($insDel)
    {
        $this->insDel = $insDel === null ? true : $insDel;
    }

    /**
     * Get Display Formatting Revisions
     *
     * @return bool True if formatting revisions are shown
     */
    public function hasFormatting()
    {
        return $this->formatting;
    }

    /**
     * Set Display Formatting Revisions
     *
     * @param bool|null $formatting
     *            Set to true to show formatting revisions
     */
    public function setFormatting($formatting = null)
    {
        $this->formatting = $formatting === null ? true : $formatting;
    }

    /**
     * Get Display Ink Annotations
     *
     * @return bool True if ink annotations are shown
     */
    public function hasInkAnnotations()
    {
        return $this->inkAnnotations;
    }

    /**
     * Set Display Ink Annotations
     *
     * @param bool $inkAnnotations
     *            Set to true to show ink annotations
     */
    public function setInkAnnotations($inkAnnotations)
    {
        $this->inkAnnotations = $inkAnnotations === null ? true : $inkAnnotations;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/AbstractContainer.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Container abstract class
 *
 * @method Text addText(string $text, mixed $fStyle = null, mixed $pStyle = null)
 * @method TextRun addTextRun(mixed $pStyle = null)
 * @method Bookmark addBookmark(string $name)
 * @method Link addLink(string $target, string $text = null, mixed $fStyle = null, mixed $pStyle = null, boolean $internal = false)
 * @method PreserveText addPreserveText(string $text, mixed $fStyle = null, mixed $pStyle = null)
 * @method void addTextBreak(int $count = 1, mixed $fStyle = null, mixed $pStyle = null)
 * @method ListItem addListItem(string $txt, int $depth = 0, mixed $font = null, mixed $list = null, mixed $para = null)
 * @method ListItemRun addListItemRun(int $depth = 0, mixed $listStyle = null, mixed $pStyle = null)
 * @method Footnote addFootnote(mixed $pStyle = null)
 * @method Endnote addEndnote(mixed $pStyle = null)
 * @method CheckBox addCheckBox(string $name, $text, mixed $fStyle = null, mixed $pStyle = null)
 * @method Title addTitle(mixed $text, int $depth = 1)
 * @method TOC addTOC(mixed $fontStyle = null, mixed $tocStyle = null, int $minDepth = 1, int $maxDepth = 9)
 * @method PageBreak addPageBreak()
 * @method Table addTable(mixed $style = null)
 * @method Image addImage(string $source, mixed $style = null, bool $isWatermark = false, $name = null)
 * @method OLEObject addOLEObject(string $source, mixed $style = null)
 * @method TextBox addTextBox(mixed $style = null)
 * @method Field addField(string $type = null, array $properties = array(), array $options = array(), mixed $text = null)
 * @method Line addLine(mixed $lineStyle = null)
 * @method Shape addShape(string $type, mixed $style = null)
 * @method Chart addChart(string $type, array $categories, array $values, array $style = null, $seriesName = null)
 * @method FormField addFormField(string $type, mixed $fStyle = null, mixed $pStyle = null)
 * @method SDT addSDT(string $type)
 *
 * @method \PhpOffice\PhpWord\Element\OLEObject addObject(string $source, mixed $style = null) deprecated, use addOLEObject instead
 *
 * @since 0.10.0
 */
abstract class AbstractContainer extends AbstractElement
{
    /**
     * Elements collection
     *
     * @var \PhpOffice\PhpWord\Element\AbstractElement[]
     */
    protected $elements = array();

    /**
     * Container type Section|Header|Footer|Footnote|Endnote|Cell|TextRun|TextBox|ListItemRun|TrackChange
     *
     * @var string
     */
    protected $container;

    /**
     * Magic method to catch all 'addElement' variation
     *
     * This removes addText, addTextRun, etc. When adding new element, we have to
     * add the model in the class docblock with `@method`.
     *
     * Warning: This makes capitalization matters, e.g. addCheckbox or addcheckbox won't work.
     *
     * @param mixed $function
     * @param mixed $args
     * @return \PhpOffice\PhpWord\Element\AbstractElement
     */
    public function __call($function, $args)
    {
        $elements = array(
            'Text', 'TextRun', 'Bookmark', 'Link', 'PreserveText', 'TextBreak',
            'ListItem', 'ListItemRun', 'Table', 'Image', 'Object', 'OLEObject',
            'Footnote', 'Endnote', 'CheckBox', 'TextBox', 'Field',
            'Line', 'Shape', 'Title', 'TOC', 'PageBreak',
            'Chart', 'FormField', 'SDT', 'Comment',
        );
        $functions = array();
        foreach ($elements as $element) {
            $functions['add' . strtolower($element)] = $element == 'Object' ? 'OLEObject' : $element;
        }

        // Run valid `add` command
        $function = strtolower($function);
        if (isset($functions[$function])) {
            $element = $functions[$function];

            // Special case for TextBreak
            // @todo Remove the `$count` parameter in 1.0.0 to make this element similiar to other elements?
            if ($element == 'TextBreak') {
                list($count, $fontStyle, $paragraphStyle) = array_pad($args, 3, null);
                if ($count === null) {
                    $count = 1;
                }
                for ($i = 1; $i <= $count; $i++) {
                    $this->addElement($element, $fontStyle, $paragraphStyle);
                }
            } else {
                // All other elements
                array_unshift($args, $element); // Prepend element name to the beginning of args array

                return call_user_func_array(array($this, 'addElement'), $args);
            }
        }

        return null;
    }

    /**
     * Add element
     *
     * Each element has different number of parameters passed
     *
     * @param string $elementName
     * @return \PhpOffice\PhpWord\Element\AbstractElement
     */
    protected function addElement($elementName)
    {
        $elementClass = __NAMESPACE__ . '\\' . $elementName;
        $this->checkValidity($elementName);

        // Get arguments
        $args = func_get_args();
        $withoutP = in_array($this->container, array('TextRun', 'Footnote', 'Endnote', 'ListItemRun', 'Field'));
        if ($withoutP && ($elementName == 'Text' || $elementName == 'PreserveText')) {
            $args[3] = null; // Remove paragraph style for texts in textrun
        }

        // Create element using reflection
        $reflection = new \ReflectionClass($elementClass);
        $elementArgs = $args;
        array_shift($elementArgs); // Shift the $elementName off the beginning of array

        /** @var \PhpOffice\PhpWord\Element\AbstractElement $element Type hint */
        $element = $reflection->newInstanceArgs($elementArgs);

        // Set parent container
        $element->setParentContainer($this);
        $element->setElementIndex($this->countElements() + 1);
        $element->setElementId();

        $this->elements[] = $element;

        return $element;
    }

    /**
     * Get all elements
     *
     * @return \PhpOffice\PhpWord\Element\AbstractElement[]
     */
    public function getElements()
    {
        return $this->elements;
    }

    /**
     * Returns the element at the requested position
     *
     * @param int $index
     * @return \PhpOffice\PhpWord\Element\AbstractElement|null
     */
    public function getElement($index)
    {
        if (array_key_exists($index, $this->elements)) {
            return $this->elements[$index];
        }

        return null;
    }

    /**
     * Removes the element at requested index
     *
     * @param int|\PhpOffice\PhpWord\Element\AbstractElement $toRemove
     */
    public function removeElement($toRemove)
    {
        if (is_int($toRemove) && array_key_exists($toRemove, $this->elements)) {
            unset($this->elements[$toRemove]);
        } elseif ($toRemove instanceof \PhpOffice\PhpWord\Element\AbstractElement) {
            foreach ($this->elements as $key => $element) {
                if ($element->getElementId() === $toRemove->getElementId()) {
                    unset($this->elements[$key]);

                    return;
                }
            }
        }
    }

    /**
     * Count elements
     *
     * @return int
     */
    public function countElements()
    {
        return count($this->elements);
    }

    /**
     * Check if a method is allowed for the current container
     *
     * @param string $method
     *
     * @throws \BadMethodCallException
     * @return bool
     */
    private function checkValidity($method)
    {
        $generalContainers = array(
            'Section', 'Header', 'Footer', 'Footnote', 'Endnote', 'Cell', 'TextRun', 'TextBox', 'ListItemRun', 'TrackChange',
        );

        $validContainers = array(
            'Text'          => $generalContainers,
            'Bookmark'      => $generalContainers,
            'Link'          => $generalContainers,
            'TextBreak'     => $generalContainers,
            'Image'         => $generalContainers,
            'OLEObject'     => $generalContainers,
            'Field'         => $generalContainers,
            'Line'          => $generalContainers,
            'Shape'         => $generalContainers,
            'FormField'     => $generalContainers,
            'SDT'           => $generalContainers,
            'TrackChange'   => $generalContainers,
            'TextRun'       => array('Section', 'Header', 'Footer', 'Cell', 'TextBox', 'TrackChange', 'ListItemRun'),
            'ListItem'      => array('Section', 'Header', 'Footer', 'Cell', 'TextBox'),
            'ListItemRun'   => array('Section', 'Header', 'Footer', 'Cell', 'TextBox'),
            'Table'         => array('Section', 'Header', 'Footer', 'Cell', 'TextBox'),
            'CheckBox'      => array('Section', 'Header', 'Footer', 'Cell', 'TextRun'),
            'TextBox'       => array('Section', 'Header', 'Footer', 'Cell'),
            'Footnote'      => array('Section', 'TextRun', 'Cell', 'ListItemRun'),
            'Endnote'       => array('Section', 'TextRun', 'Cell'),
            'PreserveText'  => array('Section', 'Header', 'Footer', 'Cell'),
            'Title'         => array('Section', 'Cell'),
            'TOC'           => array('Section'),
            'PageBreak'     => array('Section'),
            'Chart'         => array('Section', 'Cell'),
        );

        // Special condition, e.g. preservetext can only exists in cell when
        // the cell is located in header or footer
        $validSubcontainers = array(
            'PreserveText'  => array(array('Cell'), array('Header', 'Footer', 'Section')),
            'Footnote'      => array(array('Cell', 'TextRun'), array('Section')),
            'Endnote'       => array(array('Cell', 'TextRun'), array('Section')),
        );

        // Check if a method is valid for current container
        if (isset($validContainers[$method])) {
            if (!in_array($this->container, $validContainers[$method])) {
                throw new \BadMethodCallException("Cannot add {$method} in {$this->container}.");
            }
        }

        // Check if a method is valid for current container, located in other container
        if (isset($validSubcontainers[$method])) {
            $rules = $validSubcontainers[$method];
            $containers = $rules[0];
            $allowedDocParts = $rules[1];
            foreach ($containers as $container) {
                if ($this->container == $container && !in_array($this->getDocPart(), $allowedDocParts)) {
                    throw new \BadMethodCallException("Cannot add {$method} in {$this->container}.");
                }
            }
        }

        return true;
    }

    /**
     * Create textrun element
     *
     * @deprecated 0.10.0
     *
     * @param mixed $paragraphStyle
     *
     * @return \PhpOffice\PhpWord\Element\TextRun
     *
     * @codeCoverageIgnore
     */
    public function createTextRun($paragraphStyle = null)
    {
        return $this->addTextRun($paragraphStyle);
    }

    /**
     * Create footnote element
     *
     * @deprecated 0.10.0
     *
     * @param mixed $paragraphStyle
     *
     * @return \PhpOffice\PhpWord\Element\Footnote
     *
     * @codeCoverageIgnore
     */
    public function createFootnote($paragraphStyle = null)
    {
        return $this->addFootnote($paragraphStyle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/AbstractElement.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Media;
use PhpOffice\PhpWord\PhpWord;

/**
 * Element abstract class
 *
 * @since 0.10.0
 */
abstract class AbstractElement
{
    /**
     * PhpWord object
     *
     * @var \PhpOffice\PhpWord\PhpWord
     */
    protected $phpWord;

    /**
     * Section Id
     *
     * @var int
     */
    protected $sectionId;

    /**
     * Document part type: Section|Header|Footer|Footnote|Endnote
     *
     * Used by textrun and cell container to determine where the element is
     * located because it will affect the availability of other element,
     * e.g. footnote will not be available when $docPart is header or footer.
     *
     * @var string
     */
    protected $docPart = 'Section';

    /**
     * Document part Id
     *
     * For header and footer, this will be = ($sectionId - 1) * 3 + $index
     * because the max number of header/footer in every page is 3, i.e.
     * AUTO, FIRST, and EVEN (AUTO = ODD)
     *
     * @var int
     */
    protected $docPartId = 1;

    /**
     * Index of element in the elements collection (start with 1)
     *
     * @var int
     */
    protected $elementIndex = 1;

    /**
     * Unique Id for element
     *
     * @var string
     */
    protected $elementId;

    /**
     * Relation Id
     *
     * @var int
     */
    protected $relationId;

    /**
     * Depth of table container nested level; Primarily used for RTF writer/reader
     *
     * 0 = Not in a table; 1 = in a table; 2 = in a table inside another table, etc.
     *
     * @var int
     */
    private $nestedLevel = 0;

    /**
     * A reference to the parent
     *
     * @var AbstractElement|null
     */
    private $parent;

    /**
     * changed element info
     *
     * @var TrackChange
     */
    private $trackChange;

    /**
     * Parent container type
     *
     * @var string
     */
    private $parentContainer;

    /**
     * Has media relation flag; true for Link, Image, and Object
     *
     * @var bool
     */
    protected $mediaRelation = false;

    /**
     * Is part of collection; true for Title, Footnote, Endnote, Chart, and Comment
     *
     * @var bool
     */
    protected $collectionRelation = false;

    /**
     * The start position for the linked comment
     *
     * @var Comment
     */
    protected $commentRangeStart;

    /**
     * The end position for the linked comment
     *
     * @var Comment
     */
    protected $commentRangeEnd;

    /**
     * Get PhpWord
     *
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function getPhpWord()
    {
        return $this->phpWord;
    }

    /**
     * Set PhpWord as reference.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function setPhpWord(PhpWord $phpWord = null)
    {
        $this->phpWord = $phpWord;
    }

    /**
     * Get section number
     *
     * @return int
     */
    public function getSectionId()
    {
        return $this->sectionId;
    }

    /**
     * Set doc part.
     *
     * @param string $docPart
     * @param int $docPartId
     */
    public function setDocPart($docPart, $docPartId = 1)
    {
        $this->docPart = $docPart;
        $this->docPartId = $docPartId;
    }

    /**
     * Get doc part
     *
     * @return string
     */
    public function getDocPart()
    {
        return $this->docPart;
    }

    /**
     * Get doc part Id
     *
     * @return int
     */
    public function getDocPartId()
    {
        return $this->docPartId;
    }

    /**
     * Return media element (image, object, link) container name
     *
     * @return string section|headerx|footerx|footnote|endnote
     */
    private function getMediaPart()
    {
        $mediaPart = $this->docPart;
        if ($mediaPart == 'Header' || $mediaPart == 'Footer') {
            $mediaPart .= $this->docPartId;
        }

        return strtolower($mediaPart);
    }

    /**
     * Get element index
     *
     * @return int
     */
    public function getElementIndex()
    {
        return $this->elementIndex;
    }

    /**
     * Set element index.
     *
     * @param int $value
     */
    public function setElementIndex($value)
    {
        $this->elementIndex = $value;
    }

    /**
     * Get element unique ID
     *
     * @return string
     */
    public function getElementId()
    {
        return $this->elementId;
    }

    /**
     * Set element unique ID from 6 first digit of md5.
     */
    public function setElementId()
    {
        $this->elementId = substr(md5(rand()), 0, 6);
    }

    /**
     * Get relation Id
     *
     * @return int
     */
    public function getRelationId()
    {
        return $this->relationId;
    }

    /**
     * Set relation Id.
     *
     * @param int $value
     */
    public function setRelationId($value)
    {
        $this->relationId = $value;
    }

    /**
     * Get nested level
     *
     * @return int
     */
    public function getNestedLevel()
    {
        return $this->nestedLevel;
    }

    /**
     * Get comment start
     *
     * @return Comment
     */
    public function getCommentRangeStart()
    {
        return $this->commentRangeStart;
    }

    /**
     * Set comment start
     *
     * @param Comment $value
     */
    public function setCommentRangeStart(Comment $value)
    {
        if ($this instanceof Comment) {
            throw new \InvalidArgumentException('Cannot set a Comment on a Comment');
        }
        $this->commentRangeStart = $value;
        $this->commentRangeStart->setStartElement($this);
    }

    /**
     * Get comment end
     *
     * @return Comment
     */
    public function getCommentRangeEnd()
    {
        return $this->commentRangeEnd;
    }

    /**
     * Set comment end
     *
     * @param Comment $value
     */
    public function setCommentRangeEnd(Comment $value)
    {
        if ($this instanceof Comment) {
            throw new \InvalidArgumentException('Cannot set a Comment on a Comment');
        }
        $this->commentRangeEnd = $value;
        $this->commentRangeEnd->setEndElement($this);
    }

    /**
     * Get parent element
     *
     * @return AbstractElement|null
     */
    public function getParent()
    {
        return $this->parent;
    }

    /**
     * Set parent container
     *
     * Passed parameter should be a container, except for Table (contain Row) and Row (contain Cell)
     *
     * @param \PhpOffice\PhpWord\Element\AbstractElement $container
     */
    public function setParentContainer(self $container)
    {
        $this->parentContainer = substr(get_class($container), strrpos(get_class($container), '\\') + 1);
        $this->parent = $container;

        // Set nested level
        $this->nestedLevel = $container->getNestedLevel();
        if ($this->parentContainer == 'Cell') {
            $this->nestedLevel++;
        }

        // Set phpword
        $this->setPhpWord($container->getPhpWord());

        // Set doc part
        if (!$this instanceof Footnote) {
            $this->setDocPart($container->getDocPart(), $container->getDocPartId());
        }

        $this->setMediaRelation();
        $this->setCollectionRelation();
    }

    /**
     * Set relation Id for media elements (link, image, object; legacy of OOXML)
     *
     * - Image element needs to be passed to Media object
     * - Icon needs to be set for Object element
     */
    private function setMediaRelation()
    {
        if (!$this instanceof Link && !$this instanceof Image && !$this instanceof OLEObject) {
            return;
        }

        $elementName = substr(get_class($this), strrpos(get_class($this), '\\') + 1);
        if ($elementName == 'OLEObject') {
            $elementName = 'Object';
        }
        $mediaPart = $this->getMediaPart();
        $source = $this->getSource();
        $image = null;
        if ($this instanceof Image) {
            $image = $this;
        }
        $rId = Media::addElement($mediaPart, strtolower($elementName), $source, $image);
        $this->setRelationId($rId);

        if ($this instanceof OLEObject) {
            $icon = $this->getIcon();
            $rId = Media::addElement($mediaPart, 'image', $icon, new Image($icon));
            $this->setImageRelationId($rId);
        }
    }

    /**
     * Set relation Id for elements that will be registered in the Collection subnamespaces.
     */
    private function setCollectionRelation()
    {
        if ($this->collectionRelation === true && $this->phpWord instanceof PhpWord) {
            $elementName = substr(get_class($this), strrpos(get_class($this), '\\') + 1);
            $addMethod = "add{$elementName}";
            $rId = $this->phpWord->$addMethod($this);
            $this->setRelationId($rId);
        }
    }

    /**
     * Check if element is located in Section doc part (as opposed to Header/Footer)
     *
     * @return bool
     */
    public function isInSection()
    {
        return $this->docPart == 'Section';
    }

    /**
     * Set new style value
     *
     * @param mixed $styleObject Style object
     * @param mixed $styleValue Style value
     * @param bool $returnObject Always return object
     * @return mixed
     */
    protected function setNewStyle($styleObject, $styleValue = null, $returnObject = false)
    {
        if (!is_null($styleValue) && is_array($styleValue)) {
            $styleObject->setStyleByArray($styleValue);
            $style = $styleObject;
        } else {
            $style = $returnObject ? $styleObject : $styleValue;
        }

        return $style;
    }

    /**
     * Sets the trackChange information
     *
     * @param TrackChange $trackChange
     */
    public function setTrackChange(TrackChange $trackChange)
    {
        $this->trackChange = $trackChange;
    }

    /**
     * Gets the trackChange information
     *
     * @return TrackChange
     */
    public function getTrackChange()
    {
        return $this->trackChange;
    }

    /**
     * Set changed
     *
     * @param string $type INSERTED|DELETED
     * @param string $author
     * @param null|int|\DateTime $date allways in UTC
     */
    public function setChangeInfo($type, $author, $date = null)
    {
        $this->trackChange = new TrackChange($type, $author, $date);
    }

    /**
     * Set enum value
     *
     * @param string|null $value
     * @param string[] $enum
     * @param string|null $default
     *
     * @throws \InvalidArgumentException
     * @return string|null
     *
     * @todo Merge with the same method in AbstractStyle
     */
    protected function setEnumVal($value = null, $enum = array(), $default = null)
    {
        if ($value !== null && trim($value) != '' && !empty($enum) && !in_array($value, $enum)) {
            throw new \InvalidArgumentException("Invalid style value: {$value}");
        } elseif ($value === null || trim($value) == '') {
            $value = $default;
        }

        return $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Bookmark.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;

/**
 * Bookmark element
 */
class Bookmark extends AbstractElement
{
    /**
     * Bookmark Name
     *
     * @var string
     */
    private $name;

    /**
     * Is part of collection
     *
     * @var bool
     */
    protected $collectionRelation = true;

    /**
     * Create a new Bookmark Element
     *
     * @param string $name
     */
    public function __construct($name = '')
    {
        $this->name = SharedText::toUTF8($name);
    }

    /**
     * Get Bookmark name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Cell.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Cell as CellStyle;

/**
 * Table cell element
 */
class Cell extends AbstractContainer
{
    /**
     * @var string Container type
     */
    protected $container = 'Cell';

    /**
     * Cell width
     *
     * @var int
     */
    private $width = null;

    /**
     * Cell style
     *
     * @var \PhpOffice\PhpWord\Style\Cell
     */
    private $style;

    /**
     * Create new instance
     *
     * @param int $width
     * @param array|\PhpOffice\PhpWord\Style\Cell $style
     */
    public function __construct($width = null, $style = null)
    {
        $this->width = $width;
        $this->style = $this->setNewStyle(new CellStyle(), $style, true);
    }

    /**
     * Get cell style
     *
     * @return \PhpOffice\PhpWord\Style\Cell
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get cell width
     *
     * @return int
     */
    public function getWidth()
    {
        return $this->width;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Chart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Chart as ChartStyle;

/**
 * Chart element
 *
 * @since 0.12.0
 */
class Chart extends AbstractElement
{
    /**
     * Is part of collection
     *
     * @var bool
     */
    protected $collectionRelation = true;

    /**
     * Type
     *
     * @var string
     */
    private $type = 'pie';

    /**
     * Series
     *
     * @var array
     */
    private $series = array();

    /**
     * Chart style
     *
     * @var \PhpOffice\PhpWord\Style\Chart
     */
    private $style;

    /**
     * Create new instance
     *
     * @param string $type
     * @param array $categories
     * @param array $values
     * @param array $style
     * @param null|mixed $seriesName
     */
    public function __construct($type, $categories, $values, $style = null, $seriesName = null)
    {
        $this->setType($type);
        $this->addSeries($categories, $values, $seriesName);
        $this->style = $this->setNewStyle(new ChartStyle(), $style, true);
    }

    /**
     * Get type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set type.
     *
     * @param string $value
     */
    public function setType($value)
    {
        $enum = array('pie', 'doughnut', 'line', 'bar', 'stacked_bar', 'percent_stacked_bar', 'column', 'stacked_column', 'percent_stacked_column', 'area', 'radar', 'scatter');
        $this->type = $this->setEnumVal($value, $enum, 'pie');
    }

    /**
     * Add series
     *
     * @param array $categories
     * @param array $values
     * @param null|mixed $name
     */
    public function addSeries($categories, $values, $name = null)
    {
        $this->series[] = array(
            'categories' => $categories,
            'values'     => $values,
            'name'       => $name,
        );
    }

    /**
     * Get series
     *
     * @return array
     */
    public function getSeries()
    {
        return $this->series;
    }

    /**
     * Get chart style
     *
     * @return \PhpOffice\PhpWord\Style\Chart
     */
    public function getStyle()
    {
        return $this->style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/CheckBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;

/**
 * Check box element
 *
 * @since 0.10.0
 */
class CheckBox extends Text
{
    /**
     * Name content
     *
     * @var string
     */
    private $name;

    /**
     * Create new instance
     *
     * @param string $name
     * @param string $text
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($name = null, $text = null, $fontStyle = null, $paragraphStyle = null)
    {
        $this->setName($name);
        parent::__construct($text, $fontStyle, $paragraphStyle);
    }

    /**
     * Set name content
     *
     * @param string $name
     * @return self
     */
    public function setName($name)
    {
        $this->name = SharedText::toUTF8($name);

        return $this;
    }

    /**
     * Get name content
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Comment.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Comment element
 * @see http://datypic.com/sc/ooxml/t-w_CT_Comment.html
 */
class Comment extends TrackChange
{
    /**
     * Initials
     *
     * @var string
     */
    private $initials;

    /**
     * The Element where this comment starts
     *
     * @var AbstractElement
     */
    private $startElement;

    /**
     * The Element where this comment ends
     *
     * @var AbstractElement
     */
    private $endElement;

    /**
     * Is part of collection
     *
     * @var bool
     */
    protected $collectionRelation = true;

    /**
     * Create a new Comment Element
     *
     * @param string $author
     * @param null|\DateTime $date
     * @param string $initials
     */
    public function __construct($author, $date = null, $initials = null)
    {
        parent::__construct(null, $author, $date);
        $this->initials = $initials;
    }

    /**
     * Get Initials
     *
     * @return string
     */
    public function getInitials()
    {
        return $this->initials;
    }

    /**
     * Sets the element where this comment starts
     *
     * @param \PhpOffice\PhpWord\Element\AbstractElement $value
     */
    public function setStartElement(AbstractElement $value)
    {
        $this->startElement = $value;
        if ($value->getCommentRangeStart() == null) {
            $value->setCommentRangeStart($this);
        }
    }

    /**
     * Get the element where this comment starts
     *
     * @return \PhpOffice\PhpWord\Element\AbstractElement
     */
    public function getStartElement()
    {
        return $this->startElement;
    }

    /**
     * Sets the element where this comment ends
     *
     * @param \PhpOffice\PhpWord\Element\AbstractElement $value
     */
    public function setEndElement(AbstractElement $value)
    {
        $this->endElement = $value;
        if ($value->getCommentRangeEnd() == null) {
            $value->setCommentRangeEnd($this);
        }
    }

    /**
     * Get the element where this comment ends
     *
     * @return \PhpOffice\PhpWord\Element\AbstractElement
     */
    public function getEndElement()
    {
        return $this->endElement;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Endnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Endnote element
 *
 * @since 0.10.0
 */
class Endnote extends Footnote
{
    /**
     * @var string Container type
     */
    protected $container = 'Endnote';

    /**
     * Create new instance
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $paragraphStyle
     */
    public function __construct($paragraphStyle = null)
    {
        parent::__construct($paragraphStyle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Field.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Font;

/**
 * Field element
 *
 * @since 0.11.0
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_SimpleField.html
 */
class Field extends AbstractElement
{
    /**
     * Field properties and options. Depending on type, a field can have different properties
     * and options
     *
     * @var array
     */
    protected $fieldsArray = array(
        'PAGE' => array(
           'properties' => array(
               'format' => array('Arabic', 'ArabicDash', 'alphabetic', 'ALPHABETIC', 'roman', 'ROMAN'),
           ),
           'options' => array('PreserveFormat'),
        ),
        'NUMPAGES' => array(
           'properties' => array(
               'format' => array('Arabic', 'ArabicDash', 'CardText', 'DollarText', 'Ordinal', 'OrdText',
                   'alphabetic', 'ALPHABETIC', 'roman', 'ROMAN', 'Caps', 'FirstCap', 'Lower', 'Upper', ),
               'numformat' => array('0', '0,00', '#.##0', '#.##0,00', '€ #.##0,00(€ #.##0,00)', '0%', '0,00%'),
           ),
           'options' => array('PreserveFormat'),
        ),
        'DATE' => array(
            'properties' => array(
                'dateformat' => array(
                    /* Generic formats */
                    'yyyy-MM-dd', 'yyyy-MM', 'MMM-yy', 'MMM-yyyy', 'h:mm am/pm', 'h:mm:ss am/pm', 'HH:mm', 'HH:mm:ss',
                    /* Day-Month-Year formats */
                    'dddd d MMMM yyyy', 'd MMMM yyyy', 'd-MMM-yy', 'd MMM. yy',
                    'd-M-yy', 'd-M-yy h:mm', 'd-M-yy h:mm:ss', 'd-M-yy h:mm am/pm', 'd-M-yy h:mm:ss am/pm', 'd-M-yy HH:mm', 'd-M-yy HH:mm:ss',
                    'd/M/yy', 'd/M/yy h:mm', 'd/M/yy h:mm:ss', 'd/M/yy h:mm am/pm', 'd/M/yy h:mm:ss am/pm', 'd/M/yy HH:mm', 'd/M/yy HH:mm:ss',
                    'd-M-yyyy', 'd-M-yyyy h:mm', 'd-M-yyyy h:mm:ss', 'd-M-yyyy h:mm am/pm', 'd-M-yyyy h:mm:ss am/pm', 'd-M-yyyy HH:mm', 'd-M-yyyy HH:mm:ss',
                    'd/M/yyyy', 'd/M/yyyy h:mm', 'd/M/yyyy h:mm:ss', 'd/M/yyyy h:mm am/pm', 'd/M/yyyy h:mm:ss am/pm', 'd/M/yyyy HH:mm', 'd/M/yyyy HH:mm:ss',
                    /* Month-Day-Year formats */
                    'dddd, MMMM d yyyy', 'MMMM d yyyy', 'MMM-d-yy', 'MMM. d yy',
                    'M-d-yy', 'M-d-yy h:mm', 'M-d-yy h:mm:ss', 'M-d-yy h:mm am/pm', 'M-d-yy h:mm:ss am/pm', 'M-d-yy HH:mm', 'M-d-yy HH:mm:ss',
                    'M/d/yy', 'M/d/yy h:mm', 'M/d/yy h:mm:ss', 'M/d/yy h:mm am/pm', 'M/d/yy h:mm:ss am/pm', 'M/d/yy HH:mm', 'M/d/yy HH:mm:ss',
                    'M-d-yyyy', 'M-d-yyyy h:mm', 'M-d-yyyy h:mm:ss', 'M-d-yyyy h:mm am/pm', 'M-d-yyyy h:mm:ss am/pm', 'M-d-yyyy HH:mm', 'M-d-yyyy HH:mm:ss',
                    'M/d/yyyy', 'M/d/yyyy h:mm', 'M/d/yyyy h:mm:ss', 'M/d/yyyy h:mm am/pm', 'M/d/yyyy h:mm:ss am/pm', 'M/d/yyyy HH:mm', 'M/d/yyyy HH:mm:ss',
                ),
            ),
            'options' => array('PreserveFormat', 'LunarCalendar', 'SakaEraCalendar', 'LastUsedFormat'),
        ),
        'MACROBUTTON' => array(
            'properties' => array('macroname' => ''),
        ),
        'XE' => array(
            'properties' => array(),
            'options'    => array('Bold', 'Italic'),
        ),
        'INDEX' => array(
            'properties' => array(),
            'options'    => array('PreserveFormat'),
        ),
        'STYLEREF' => array(
            'properties' => array('StyleIdentifier' => ''),
            'options'    => array('PreserveFormat'),
        ),
    );

    /**
     * Field type
     *
     * @var string
     */
    protected $type;

    /**
     * Field text
     *
     * @var TextRun|string
     */
    protected $text;

    /**
     * Field properties
     *
     * @var array
     */
    protected $properties = array();

    /**
     * Field options
     *
     * @var array
     */
    protected $options = array();

    /**
     * Font style
     *
     * @var string|\PhpOffice\PhpWord\Style\Font
     */
    protected $fontStyle;

    /**
     * Set Font style
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Font $style
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function setFontStyle($style = null)
    {
        if ($style instanceof Font) {
            $this->fontStyle = $style;
        } elseif (is_array($style)) {
            $this->fontStyle = new Font('text');
            $this->fontStyle->setStyleByArray($style);
        } elseif (null === $style) {
            $this->fontStyle = null;
        } else {
            $this->fontStyle = $style;
        }

        return $this->fontStyle;
    }

    /**
     * Get Font style
     *
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function getFontStyle()
    {
        return $this->fontStyle;
    }

    /**
     * Create a new Field Element
     *
     * @param string $type
     * @param array $properties
     * @param array $options
     * @param TextRun|string|null $text
     * @param string|array|\PhpOffice\PhpWord\Style\Font $fontStyle
     */
    public function __construct($type = null, $properties = array(), $options = array(), $text = null, $fontStyle = null)
    {
        $this->setType($type);
        $this->setProperties($properties);
        $this->setOptions($options);
        $this->setText($text);
        $this->setFontStyle($fontStyle);
    }

    /**
     * Set Field type
     *
     * @param string $type
     *
     * @throws \InvalidArgumentException
     * @return string
     */
    public function setType($type = null)
    {
        if (isset($type)) {
            if (isset($this->fieldsArray[$type])) {
                $this->type = $type;
            } else {
                throw new \InvalidArgumentException("Invalid type '$type'");
            }
        }

        return $this->type;
    }

    /**
     * Get Field type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set Field properties
     *
     * @param array $properties
     *
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setProperties($properties = array())
    {
        if (is_array($properties)) {
            foreach (array_keys($properties) as $propkey) {
                if (!(isset($this->fieldsArray[$this->type]['properties'][$propkey]))) {
                    throw new \InvalidArgumentException("Invalid property '$propkey'");
                }
            }
            $this->properties = array_merge($this->properties, $properties);
        }

        return $this->properties;
    }

    /**
     * Get Field properties
     *
     * @return array
     */
    public function getProperties()
    {
        return $this->properties;
    }

    /**
     * Set Field options
     *
     * @param array $options
     *
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setOptions($options = array())
    {
        if (is_array($options)) {
            foreach (array_keys($options) as $optionkey) {
                if (!(isset($this->fieldsArray[$this->type]['options'][$optionkey])) && substr($optionkey, 0, 1) !== '\\') {
                    throw new \InvalidArgumentException("Invalid option '$optionkey', possible values are " . implode(', ', $this->fieldsArray[$this->type]['options']));
                }
            }
            $this->options = array_merge($this->options, $options);
        }

        return $this->options;
    }

    /**
     * Get Field properties
     *
     * @return array
     */
    public function getOptions()
    {
        return $this->options;
    }

    /**
     * Set Field text
     *
     * @param string|TextRun $text
     *
     * @throws \InvalidArgumentException
     * @return null|string|TextRun
     */
    public function setText($text = null)
    {
        if (isset($text)) {
            if (is_string($text) || $text instanceof TextRun) {
                $this->text = $text;
            } else {
                throw new \InvalidArgumentException('Invalid text');
            }
        }

        return $this->text;
    }

    /**
     * Get Field text
     *
     * @return string|TextRun
     */
    public function getText()
    {
        return $this->text;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Footer.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Footer element
 */
class Footer extends AbstractContainer
{
    /**
     * Header/footer types constants
     *
     * @var string
     * @see  http://www.datypic.com/sc/ooxml/t-w_ST_HdrFtr.html Header or Footer Type
     */
    const AUTO = 'default';  // default and odd pages
    const FIRST = 'first';
    const EVEN = 'even';

    /**
     * @var string Container type
     */
    protected $container = 'Footer';

    /**
     * Header type
     *
     * @var string
     */
    protected $type = self::AUTO;

    /**
     * Create new instance
     *
     * @param int $sectionId
     * @param int $containerId
     * @param string $type
     */
    public function __construct($sectionId, $containerId = 1, $type = self::AUTO)
    {
        $this->sectionId = $sectionId;
        $this->setType($type);
        $this->setDocPart($this->container, ($sectionId - 1) * 3 + $containerId);
    }

    /**
     * Set type.
     *
     * @since 0.10.0
     *
     * @param string $value
     */
    public function setType($value = self::AUTO)
    {
        if (!in_array($value, array(self::AUTO, self::FIRST, self::EVEN))) {
            $value = self::AUTO;
        }
        $this->type = $value;
    }

    /**
     * Get type
     *
     * @return string
     * @since 0.10.0
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Reset type to default
     *
     * @return string
     */
    public function resetType()
    {
        return $this->type = self::AUTO;
    }

    /**
     * First page only header
     *
     * @return string
     */
    public function firstPage()
    {
        return $this->type = self::FIRST;
    }

    /**
     * Even numbered pages only
     *
     * @return string
     */
    public function evenPage()
    {
        return $this->type = self::EVEN;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Footnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Paragraph;

class Footnote extends AbstractContainer
{
    /**
     * @var string Container type
     */
    protected $container = 'Footnote';

    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    protected $paragraphStyle;

    /**
     * Is part of collection
     *
     * @var bool
     */
    protected $collectionRelation = true;

    /**
     * Create new instance
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $paragraphStyle
     */
    public function __construct($paragraphStyle = null)
    {
        $this->paragraphStyle = $this->setNewStyle(new Paragraph(), $paragraphStyle);
        $this->setDocPart($this->container);
    }

    /**
     * Get paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Get Footnote Reference ID
     *
     * @deprecated 0.10.0
     * @codeCoverageIgnore
     *
     * @return int
     */
    public function getReferenceId()
    {
        return $this->getRelationId();
    }

    /**
     * Set Footnote Reference ID
     *
     * @deprecated 0.10.0
     * @codeCoverageIgnore
     *
     * @param int $rId
     */
    public function setReferenceId($rId)
    {
        $this->setRelationId($rId);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/FormField.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Form field element
 *
 * @since 0.12.0
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_FFData.html
 */
class FormField extends Text
{
    /**
     * Form field type: textinput|checkbox|dropdown
     *
     * @var string
     */
    private $type = 'textinput';

    /**
     * Form field name
     *
     * @var string|bool|int
     */
    private $name;

    /**
     * Default value
     *
     * - TextInput: string
     * - CheckBox: bool
     * - DropDown: int Index of entries (zero based)
     *
     * @var string|bool|int
     */
    private $default;

    /**
     * Value
     *
     * @var string|bool|int
     */
    private $value;

    /**
     * Dropdown entries
     *
     * @var array
     */
    private $entries = array();

    /**
     * Create new instance
     *
     * @param string $type
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($type, $fontStyle = null, $paragraphStyle = null)
    {
        parent::__construct(null, $fontStyle, $paragraphStyle);
        $this->setType($type);
    }

    /**
     * Get type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set type
     *
     * @param string $value
     * @return self
     */
    public function setType($value)
    {
        $enum = array('textinput', 'checkbox', 'dropdown');
        $this->type = $this->setEnumVal($value, $enum, $this->type);

        return $this;
    }

    /**
     * Get name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set name
     *
     * @param string|bool|int $value
     * @return self
     */
    public function setName($value)
    {
        $this->name = $value;

        return $this;
    }

    /**
     * Get default
     *
     * @return string|bool|int
     */
    public function getDefault()
    {
        return $this->default;
    }

    /**
     * Set default
     *
     * @param string|bool|int $value
     * @return self
     */
    public function setDefault($value)
    {
        $this->default = $value;

        return $this;
    }

    /**
     * Get value
     *
     * @return string|bool|int
     */
    public function getValue()
    {
        return $this->value;
    }

    /**
     * Set value
     *
     * @param string|bool|int $value
     * @return self
     */
    public function setValue($value)
    {
        $this->value = $value;

        return $this;
    }

    /**
     * Get entries
     *
     * @return array
     */
    public function getEntries()
    {
        return $this->entries;
    }

    /**
     * Set entries
     *
     * @param array $value
     * @return self
     */
    public function setEntries($value)
    {
        $this->entries = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Header.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Header element
 */
class Header extends Footer
{
    /**
     * @var string Container type
     */
    protected $container = 'Header';

    /**
     * Add a Watermark Element
     *
     * @param string $src
     * @param mixed $style
     * @return Image
     */
    public function addWatermark($src, $style = null)
    {
        return $this->addImage($src, $style, true);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Exception\CreateTemporaryFileException;
use PhpOffice\PhpWord\Exception\InvalidImageException;
use PhpOffice\PhpWord\Exception\UnsupportedImageTypeException;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\ZipArchive;
use PhpOffice\PhpWord\Style\Image as ImageStyle;

/**
 * Image element
 */
class Image extends AbstractElement
{
    /**
     * Image source type constants
     */
    const SOURCE_LOCAL = 'local'; // Local images
    const SOURCE_GD = 'gd'; // Generated using GD
    const SOURCE_ARCHIVE = 'archive'; // Image in archives zip://$archive#$image
    const SOURCE_STRING = 'string'; // Image from string

    /**
     * Image source
     *
     * @var string
     */
    private $source;

    /**
     * Source type: local|gd|archive
     *
     * @var string
     */
    private $sourceType;

    /**
     * Image style
     *
     * @var ImageStyle
     */
    private $style;

    /**
     * Is watermark
     *
     * @var bool
     */
    private $watermark;

    /**
     * Name of image
     *
     * @var string
     */
    private $name;

    /**
     * Image type
     *
     * @var string
     */
    private $imageType;

    /**
     * Image create function
     *
     * @var string
     */
    private $imageCreateFunc;

    /**
     * Image function
     *
     * @var string
     */
    private $imageFunc;

    /**
     * Image extension
     *
     * @var string
     */
    private $imageExtension;

    /**
     * Is memory image
     *
     * @var bool
     */
    private $memoryImage;

    /**
     * Image target file name
     *
     * @var string
     */
    private $target;

    /**
     * Image media index
     *
     * @var int
     */
    private $mediaIndex;

    /**
     * Has media relation flag; true for Link, Image, and Object
     *
     * @var bool
     */
    protected $mediaRelation = true;

    /**
     * Create new image element
     *
     * @param string $source
     * @param mixed $style
     * @param bool $watermark
     * @param string $name
     *
     * @throws \PhpOffice\PhpWord\Exception\InvalidImageException
     * @throws \PhpOffice\PhpWord\Exception\UnsupportedImageTypeException
     */
    public function __construct($source, $style = null, $watermark = false, $name = null)
    {
        $this->source = $source;
        $this->style = $this->setNewStyle(new ImageStyle(), $style, true);
        $this->setIsWatermark($watermark);
        $this->setName($name);

        $this->checkImage();
    }

    /**
     * Get Image style
     *
     * @return ImageStyle
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get image source
     *
     * @return string
     */
    public function getSource()
    {
        return $this->source;
    }

    /**
     * Get image source type
     *
     * @return string
     */
    public function getSourceType()
    {
        return $this->sourceType;
    }

    /**
     * Sets the image name
     *
     * @param string $value
     */
    public function setName($value)
    {
        $this->name = $value;
    }

    /**
     * Get image name
     *
     * @return null|string
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Get image media ID
     *
     * @return string
     */
    public function getMediaId()
    {
        return md5($this->source);
    }

    /**
     * Get is watermark
     *
     * @return bool
     */
    public function isWatermark()
    {
        return $this->watermark;
    }

    /**
     * Set is watermark
     *
     * @param bool $value
     */
    public function setIsWatermark($value)
    {
        $this->watermark = $value;
    }

    /**
     * Get image type
     *
     * @return string
     */
    public function getImageType()
    {
        return $this->imageType;
    }

    /**
     * Get image create function
     *
     * @return string
     */
    public function getImageCreateFunction()
    {
        return $this->imageCreateFunc;
    }

    /**
     * Get image function
     *
     * @return string
     */
    public function getImageFunction()
    {
        return $this->imageFunc;
    }

    /**
     * Get image extension
     *
     * @return string
     */
    public function getImageExtension()
    {
        return $this->imageExtension;
    }

    /**
     * Get is memory image
     *
     * @return bool
     */
    public function isMemImage()
    {
        return $this->memoryImage;
    }

    /**
     * Get target file name
     *
     * @return string
     */
    public function getTarget()
    {
        return $this->target;
    }

    /**
     * Set target file name.
     *
     * @param string $value
     */
    public function setTarget($value)
    {
        $this->target = $value;
    }

    /**
     * Get media index
     *
     * @return int
     */
    public function getMediaIndex()
    {
        return $this->mediaIndex;
    }

    /**
     * Set media index.
     *
     * @param int $value
     */
    public function setMediaIndex($value)
    {
        $this->mediaIndex = $value;
    }

    /**
     * Get image string data
     *
     * @param bool $base64
     * @return string|null
     * @since 0.11.0
     */
    public function getImageStringData($base64 = false)
    {
        $source = $this->source;
        $actualSource = null;
        $imageBinary = null;
        $imageData = null;
        $isTemp = false;

        // Get actual source from archive image or other source
        // Return null if not found
        if ($this->sourceType == self::SOURCE_ARCHIVE) {
            $source = substr($source, 6);
            list($zipFilename, $imageFilename) = explode('#', $source);

            $zip = new ZipArchive();
            if ($zip->open($zipFilename) !== false) {
                if ($zip->locateName($imageFilename) !== false) {
                    $isTemp = true;
                    $zip->extractTo(Settings::getTempDir(), $imageFilename);
                    $actualSource = Settings::getTempDir() . DIRECTORY_SEPARATOR . $imageFilename;
                }
            }
            $zip->close();
        } else {
            $actualSource = $source;
        }

        // Can't find any case where $actualSource = null hasn't captured by
        // preceding exceptions. Please uncomment when you find the case and
        // put the case into Element\ImageTest.
        // if ($actualSource === null) {
        //     return null;
        // }

        // Read image binary data and convert to hex/base64 string
        if ($this->sourceType == self::SOURCE_GD) {
            $imageResource = call_user_func($this->imageCreateFunc, $actualSource);
            if ($this->imageType === 'image/png') {
                // PNG images need to preserve alpha channel information
                imagesavealpha($imageResource, true);
            }
            ob_start();
            call_user_func($this->imageFunc, $imageResource);
            $imageBinary = ob_get_contents();
            ob_end_clean();
        } elseif ($this->sourceType == self::SOURCE_STRING) {
            $imageBinary = $this->source;
        } else {
            $fileHandle = fopen($actualSource, 'rb', false);
            if ($fileHandle !== false) {
                $imageBinary = fread($fileHandle, filesize($actualSource));
                fclose($fileHandle);
            }
        }
        if ($imageBinary !== null) {
            if ($base64) {
                $imageData = chunk_split(base64_encode($imageBinary));
            } else {
                $imageData = chunk_split(bin2hex($imageBinary));
            }
        }

        // Delete temporary file if necessary
        if ($isTemp === true) {
            @unlink($actualSource);
        }

        return $imageData;
    }

    /**
     * Check memory image, supported type, image functions, and proportional width/height.
     *
     * @throws \PhpOffice\PhpWord\Exception\InvalidImageException
     * @throws \PhpOffice\PhpWord\Exception\UnsupportedImageTypeException
     */
    private function checkImage()
    {
        $this->setSourceType();

        // Check image data
        if ($this->sourceType == self::SOURCE_ARCHIVE) {
            $imageData = $this->getArchiveImageSize($this->source);
        } elseif ($this->sourceType == self::SOURCE_STRING) {
            $imageData = $this->getStringImageSize($this->source);
        } else {
            $imageData = @getimagesize($this->source);
        }
        if (!is_array($imageData)) {
            throw new InvalidImageException(sprintf('Invalid image: %s', $this->source));
        }
        list($actualWidth, $actualHeight, $imageType) = $imageData;

        // Check image type support
        $supportedTypes = array(IMAGETYPE_JPEG, IMAGETYPE_GIF, IMAGETYPE_PNG);
        if ($this->sourceType != self::SOURCE_GD && $this->sourceType != self::SOURCE_STRING) {
            $supportedTypes = array_merge($supportedTypes, array(IMAGETYPE_BMP, IMAGETYPE_TIFF_II, IMAGETYPE_TIFF_MM));
        }
        if (!in_array($imageType, $supportedTypes)) {
            throw new UnsupportedImageTypeException();
        }

        // Define image functions
        $this->imageType = image_type_to_mime_type($imageType);
        $this->setFunctions();
        $this->setProportionalSize($actualWidth, $actualHeight);
    }

    /**
     * Set source type.
     */
    private function setSourceType()
    {
        if (stripos(strrev($this->source), strrev('.php')) === 0) {
            $this->memoryImage = true;
            $this->sourceType = self::SOURCE_GD;
        } elseif (strpos($this->source, 'zip://') !== false) {
            $this->memoryImage = false;
            $this->sourceType = self::SOURCE_ARCHIVE;
        } elseif (filter_var($this->source, FILTER_VALIDATE_URL) !== false) {
            $this->memoryImage = true;
            if (strpos($this->source, 'https') === 0) {
                $fileContent = file_get_contents($this->source);
                $this->source = $fileContent;
                $this->sourceType = self::SOURCE_STRING;
            } else {
                $this->sourceType = self::SOURCE_GD;
            }
        } elseif ((strpos($this->source, chr(0)) === false) && @file_exists($this->source)) {
            $this->memoryImage = false;
            $this->sourceType = self::SOURCE_LOCAL;
        } else {
            $this->memoryImage = true;
            $this->sourceType = self::SOURCE_STRING;
        }
    }

    /**
     * Get image size from archive
     *
     * @since 0.12.0 Throws CreateTemporaryFileException.
     *
     * @param string $source
     *
     * @throws \PhpOffice\PhpWord\Exception\CreateTemporaryFileException
     *
     * @return array|null
     */
    private function getArchiveImageSize($source)
    {
        $imageData = null;
        $source = substr($source, 6);
        list($zipFilename, $imageFilename) = explode('#', $source);

        $tempFilename = tempnam(Settings::getTempDir(), 'PHPWordImage');
        if (false === $tempFilename) {
            throw new CreateTemporaryFileException(); // @codeCoverageIgnore
        }

        $zip = new ZipArchive();
        if ($zip->open($zipFilename) !== false) {
            if ($zip->locateName($imageFilename) !== false) {
                $imageContent = $zip->getFromName($imageFilename);
                if ($imageContent !== false) {
                    file_put_contents($tempFilename, $imageContent);
                    $imageData = getimagesize($tempFilename);
                    unlink($tempFilename);
                }
            }
            $zip->close();
        }

        return $imageData;
    }

    /**
     * get image size from string
     *
     * @param string $source
     *
     * @codeCoverageIgnore this method is just a replacement for getimagesizefromstring which exists only as of PHP 5.4
     */
    private function getStringImageSize($source)
    {
        $result = false;
        if (!function_exists('getimagesizefromstring')) {
            $uri = 'data://application/octet-stream;base64,' . base64_encode($source);
            $result = @getimagesize($uri);
        } else {
            $result = @getimagesizefromstring($source);
        }

        return $result;
    }

    /**
     * Set image functions and extensions.
     */
    private function setFunctions()
    {
        switch ($this->imageType) {
            case 'image/png':
                $this->imageCreateFunc = $this->sourceType == self::SOURCE_STRING ? 'imagecreatefromstring' : 'imagecreatefrompng';
                $this->imageFunc = 'imagepng';
                $this->imageExtension = 'png';
                break;
            case 'image/gif':
                $this->imageCreateFunc = $this->sourceType == self::SOURCE_STRING ? 'imagecreatefromstring' : 'imagecreatefromgif';
                $this->imageFunc = 'imagegif';
                $this->imageExtension = 'gif';
                break;
            case 'image/jpeg':
            case 'image/jpg':
                $this->imageCreateFunc = $this->sourceType == self::SOURCE_STRING ? 'imagecreatefromstring' : 'imagecreatefromjpeg';
                $this->imageFunc = 'imagejpeg';
                $this->imageExtension = 'jpg';
                break;
            case 'image/bmp':
            case 'image/x-ms-bmp':
                $this->imageType = 'image/bmp';
                $this->imageExtension = 'bmp';
                break;
            case 'image/tiff':
                $this->imageExtension = 'tif';
                break;
        }
    }

    /**
     * Set proportional width/height if one dimension not available.
     *
     * @param int $actualWidth
     * @param int $actualHeight
     */
    private function setProportionalSize($actualWidth, $actualHeight)
    {
        $styleWidth = $this->style->getWidth();
        $styleHeight = $this->style->getHeight();
        if (!($styleWidth && $styleHeight)) {
            if ($styleWidth == null && $styleHeight == null) {
                $this->style->setWidth($actualWidth);
                $this->style->setHeight($actualHeight);
            } elseif ($styleWidth) {
                $this->style->setHeight($actualHeight * ($styleWidth / $actualWidth));
            } else {
                $this->style->setWidth($actualWidth * ($styleHeight / $actualHeight));
            }
        }
    }

    /**
     * Get is watermark
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getIsWatermark()
    {
        return $this->isWatermark();
    }

    /**
     * Get is memory image
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getIsMemImage()
    {
        return $this->isMemImage();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Line.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Line as LineStyle;

/**
 * Line element
 */
class Line extends AbstractElement
{
    /**
     * Line style
     *
     * @var \PhpOffice\PhpWord\Style\Line
     */
    private $style;

    /**
     * Create new line element
     *
     * @param mixed $style
     */
    public function __construct($style = null)
    {
        $this->style = $this->setNewStyle(new LineStyle(), $style);
    }

    /**
     * Get line style
     *
     * @return \PhpOffice\PhpWord\Style\Line
     */
    public function getStyle()
    {
        return $this->style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Link.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;

/**
 * Link element
 */
class Link extends AbstractElement
{
    /**
     * Link source
     *
     * @var string
     */
    private $source;

    /**
     * Link text
     *
     * @var string
     */
    private $text;

    /**
     * Font style
     *
     * @var string|\PhpOffice\PhpWord\Style\Font
     */
    private $fontStyle;

    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    private $paragraphStyle;

    /**
     * Has media relation flag; true for Link, Image, and Object
     *
     * @var bool
     */
    protected $mediaRelation = true;

    /**
     * Has internal flag - anchor to internal bookmark
     *
     * @var bool
     */
    protected $internal = false;

    /**
     * Create a new Link Element
     *
     * @param string $source
     * @param string $text
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     * @param bool $internal
     */
    public function __construct($source, $text = null, $fontStyle = null, $paragraphStyle = null, $internal = false)
    {
        $this->source = SharedText::toUTF8($source);
        $this->text = is_null($text) ? $this->source : SharedText::toUTF8($text);
        $this->fontStyle = $this->setNewStyle(new Font('text'), $fontStyle);
        $this->paragraphStyle = $this->setNewStyle(new Paragraph(), $paragraphStyle);
        $this->internal = $internal;
    }

    /**
     * Get link source
     *
     * @return string
     */
    public function getSource()
    {
        return $this->source;
    }

    /**
     * Get link text
     *
     * @return string
     */
    public function getText()
    {
        return $this->text;
    }

    /**
     * Get Text style
     *
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function getFontStyle()
    {
        return $this->fontStyle;
    }

    /**
     * Get Paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Get link target
     *
     * @deprecated 0.12.0
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getTarget()
    {
        return $this->source;
    }

    /**
     * Get Link source
     *
     * @deprecated 0.10.0
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getLinkSrc()
    {
        return $this->getSource();
    }

    /**
     * Get Link name
     *
     * @deprecated 0.10.0
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getLinkName()
    {
        return $this->getText();
    }

    /**
     * is internal
     *
     * @return bool
     */
    public function isInternal()
    {
        return $this->internal;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/ListItem.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style\ListItem as ListItemStyle;

/**
 * List item element
 */
class ListItem extends AbstractElement
{
    /**
     * Element style
     *
     * @var \PhpOffice\PhpWord\Style\ListItem
     */
    private $style;

    /**
     * Text object
     *
     * @var \PhpOffice\PhpWord\Element\Text
     */
    private $textObject;

    /**
     * Depth
     *
     * @var int
     */
    private $depth;

    /**
     * Create a new ListItem
     *
     * @param string $text
     * @param int $depth
     * @param mixed $fontStyle
     * @param array|string|null $listStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($text, $depth = 0, $fontStyle = null, $listStyle = null, $paragraphStyle = null)
    {
        $this->textObject = new Text(SharedText::toUTF8($text), $fontStyle, $paragraphStyle);
        $this->depth = $depth;

        // Version >= 0.10.0 will pass numbering style name. Older version will use old method
        if (!is_null($listStyle) && is_string($listStyle)) {
            $this->style = new ListItemStyle($listStyle); // @codeCoverageIgnore
        } else {
            $this->style = $this->setNewStyle(new ListItemStyle(), $listStyle, true);
        }
    }

    /**
     * Get style
     *
     * @return \PhpOffice\PhpWord\Style\ListItem
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get Text object
     *
     * @return \PhpOffice\PhpWord\Element\Text
     */
    public function getTextObject()
    {
        return $this->textObject;
    }

    /**
     * Get depth
     *
     * @return int
     */
    public function getDepth()
    {
        return $this->depth;
    }

    /**
     * Get text
     *
     * @return string
     * @since 0.11.0
     */
    public function getText()
    {
        return $this->textObject->getText();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/ListItemRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
* word processing documents.
*
* PHPWord is free software distributed under the terms of the GNU Lesser
* General Public License version 3 as published by the Free Software Foundation.
*
* For the full copyright and license information, please read the LICENSE
* file that was distributed with this source code. For the full list of
* contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
*
* @see         https://github.com/PHPOffice/PHPWord
* @copyright   2010-2018 PHPWord contributors
* @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
*/

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\ListItem as ListItemStyle;

/**
 * List item element
 */
class ListItemRun extends TextRun
{
    /**
     * @var string Container type
     */
    protected $container = 'ListItemRun';

    /**
     * ListItem Style
     *
     * @var \PhpOffice\PhpWord\Style\ListItem
     */
    private $style;

    /**
     * ListItem Depth
     *
     * @var int
     */
    private $depth;

    /**
     * Create a new ListItem
     *
     * @param int $depth
     * @param array|string|null $listStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($depth = 0, $listStyle = null, $paragraphStyle = null)
    {
        $this->depth = $depth;

        // Version >= 0.10.0 will pass numbering style name. Older version will use old method
        if (!is_null($listStyle) && is_string($listStyle)) {
            $this->style = new ListItemStyle($listStyle);
        } else {
            $this->style = $this->setNewStyle(new ListItemStyle(), $listStyle, true);
        }
        parent::__construct($paragraphStyle);
    }

    /**
     * Get ListItem style.
     *
     * @return \PhpOffice\PhpWord\Style\ListItem
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get ListItem depth.
     *
     * @return int
     */
    public function getDepth()
    {
        return $this->depth;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/OLEObject.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Exception\InvalidObjectException;
use PhpOffice\PhpWord\Style\Image as ImageStyle;

/**
 * OLEObject element
 */
class OLEObject extends AbstractElement
{
    /**
     * Ole-Object Src
     *
     * @var string
     */
    private $source;

    /**
     * Image Style
     *
     * @var \PhpOffice\PhpWord\Style\Image
     */
    private $style;

    /**
     * Icon
     *
     * @var string
     */
    private $icon;

    /**
     * Image Relation ID
     *
     * @var int
     */
    private $imageRelationId;

    /**
     * Has media relation flag; true for Link, Image, and Object
     *
     * @var bool
     */
    protected $mediaRelation = true;

    /**
     * Create a new Ole-Object Element
     *
     * @param string $source
     * @param mixed $style
     *
     * @throws \PhpOffice\PhpWord\Exception\InvalidObjectException
     */
    public function __construct($source, $style = null)
    {
        $supportedTypes = array('xls', 'doc', 'ppt', 'xlsx', 'docx', 'pptx');
        $pathInfo = pathinfo($source);

        if (file_exists($source) && in_array($pathInfo['extension'], $supportedTypes)) {
            $ext = $pathInfo['extension'];
            if (strlen($ext) == 4 && strtolower(substr($ext, -1)) == 'x') {
                $ext = substr($ext, 0, -1);
            }

            $this->source = $source;
            $this->style = $this->setNewStyle(new ImageStyle(), $style, true);
            $this->icon = realpath(__DIR__ . "/../resources/{$ext}.png");

            return;
        }

        throw new InvalidObjectException();
    }

    /**
     * Get object source
     *
     * @return string
     */
    public function getSource()
    {
        return $this->source;
    }

    /**
     * Get object style
     *
     * @return \PhpOffice\PhpWord\Style\Image
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get object icon
     *
     * @return string
     */
    public function getIcon()
    {
        return $this->icon;
    }

    /**
     * Get image relation ID
     *
     * @return int
     */
    public function getImageRelationId()
    {
        return $this->imageRelationId;
    }

    /**
     * Set Image Relation ID.
     *
     * @param int $rId
     */
    public function setImageRelationId($rId)
    {
        $this->imageRelationId = $rId;
    }

    /**
     * Get Object ID
     *
     * @deprecated 0.10.0
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public function getObjectId()
    {
        return $this->relationId + 1325353440;
    }

    /**
     * Set Object ID
     *
     * @deprecated 0.10.0
     *
     * @param int $objId
     *
     * @codeCoverageIgnore
     */
    public function setObjectId($objId)
    {
        $this->relationId = $objId;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/PageBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Page break element
 */
class PageBreak extends AbstractElement
{
    /**
     * Create new page break
     */
    public function __construct()
    {
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/PreserveText.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;

/**
 * Preserve text/field element
 */
class PreserveText extends AbstractElement
{
    /**
     * Text content
     *
     * @var string|array
     */
    private $text;

    /**
     * Text style
     *
     * @var string|\PhpOffice\PhpWord\Style\Font
     */
    private $fontStyle;

    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    private $paragraphStyle;

    /**
     * Create a new Preserve Text Element
     *
     * @param string $text
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($text = null, $fontStyle = null, $paragraphStyle = null)
    {
        $this->fontStyle = $this->setNewStyle(new Font('text'), $fontStyle);
        $this->paragraphStyle = $this->setNewStyle(new Paragraph(), $paragraphStyle);

        $this->text = SharedText::toUTF8($text);
        $matches = preg_split('/({.*?})/', $this->text, null, PREG_SPLIT_DELIM_CAPTURE | PREG_SPLIT_NO_EMPTY);
        if (isset($matches[0])) {
            $this->text = $matches;
        }
    }

    /**
     * Get Text style
     *
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function getFontStyle()
    {
        return $this->fontStyle;
    }

    /**
     * Get Paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Get Text content
     *
     * @return string|array
     */
    public function getText()
    {
        return $this->text;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Row.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Row as RowStyle;

/**
 * Table row element
 *
 * @since 0.8.0
 */
class Row extends AbstractElement
{
    /**
     * Row height
     *
     * @var int
     */
    private $height = null;

    /**
     * Row style
     *
     * @var \PhpOffice\PhpWord\Style\Row
     */
    private $style;

    /**
     * Row cells
     *
     * @var \PhpOffice\PhpWord\Element\Cell[]
     */
    private $cells = array();

    /**
     * Create a new table row
     *
     * @param int $height
     * @param mixed $style
     */
    public function __construct($height = null, $style = null)
    {
        $this->height = $height;
        $this->style = $this->setNewStyle(new RowStyle(), $style, true);
    }

    /**
     * Add a cell
     *
     * @param int $width
     * @param mixed $style
     * @return \PhpOffice\PhpWord\Element\Cell
     */
    public function addCell($width = null, $style = null)
    {
        $cell = new Cell($width, $style);
        $cell->setParentContainer($this);
        $this->cells[] = $cell;

        return $cell;
    }

    /**
     * Get all cells
     *
     * @return \PhpOffice\PhpWord\Element\Cell[]
     */
    public function getCells()
    {
        return $this->cells;
    }

    /**
     * Get row style
     *
     * @return \PhpOffice\PhpWord\Style\Row
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get row height
     *
     * @return int
     */
    public function getHeight()
    {
        return $this->height;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/SDT.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * Structured document tag (SDT) element
 *
 * @since 0.12.0
 */
class SDT extends Text
{
    /**
     * Form field type: comboBox|dropDownList|date
     *
     * @var string
     */
    private $type;

    /**
     * Value
     *
     * @var string|bool|int
     */
    private $value;

    /**
     * CheckBox/DropDown list entries
     *
     * @var array
     */
    private $listItems = array();

    /**
     * Alias
     *
     * @var string
     */
    private $alias;

    /**
     * Tag
     *
     * @var string
     */
    private $tag;

    /**
     * Create new instance
     *
     * @param string $type
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($type, $fontStyle = null, $paragraphStyle = null)
    {
        parent::__construct(null, $fontStyle, $paragraphStyle);
        $this->setType($type);
    }

    /**
     * Get type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set type
     *
     * @param string $value
     * @return self
     */
    public function setType($value)
    {
        $enum = array('plainText', 'comboBox', 'dropDownList', 'date');
        $this->type = $this->setEnumVal($value, $enum, 'comboBox');

        return $this;
    }

    /**
     * Get value
     *
     * @return string|bool|int
     */
    public function getValue()
    {
        return $this->value;
    }

    /**
     * Set value
     *
     * @param string|bool|int $value
     * @return self
     */
    public function setValue($value)
    {
        $this->value = $value;

        return $this;
    }

    /**
     * Get listItems
     *
     * @return array
     */
    public function getListItems()
    {
        return $this->listItems;
    }

    /**
     * Set listItems
     *
     * @param array $value
     * @return self
     */
    public function setListItems($value)
    {
        $this->listItems = $value;

        return $this;
    }

    /**
     * Get tag
     *
     * @return string
     */
    public function getTag()
    {
        return $this->tag;
    }

    /**
     * Set tag
     *
     * @param string $tag
     * @return self
     */
    public function setTag($tag)
    {
        $this->tag = $tag;

        return $this;
    }

    /**
     * Get alias
     *
     * @return string
     */
    public function getAlias()
    {
        return $this->alias;
    }

    /**
     * Set alias
     *
     * @param string $alias
     * @return self
     */
    public function setAlias($alias)
    {
        $this->alias = $alias;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Section.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\ComplexType\FootnoteProperties;
use PhpOffice\PhpWord\Style\Section as SectionStyle;

class Section extends AbstractContainer
{
    /**
     * @var string Container type
     */
    protected $container = 'Section';

    /**
     * Section style
     *
     * @var \PhpOffice\PhpWord\Style\Section
     */
    private $style;

    /**
     * Section headers, indexed from 1, not zero
     *
     * @var Header[]
     */
    private $headers = array();

    /**
     * Section footers, indexed from 1, not zero
     *
     * @var Footer[]
     */
    private $footers = array();

    /**
     * The properties for the footnote of this section
     *
     * @var FootnoteProperties
     */
    private $footnoteProperties;

    /**
     * Create new instance
     *
     * @param int $sectionCount
     * @param null|array|\PhpOffice\PhpWord\Style $style
     */
    public function __construct($sectionCount, $style = null)
    {
        $this->sectionId = $sectionCount;
        $this->setDocPart($this->container, $this->sectionId);
        if (null === $style) {
            $style = new SectionStyle();
        }
        $this->style = $this->setNewStyle(new SectionStyle(), $style);
    }

    /**
     * Set section style.
     *
     * @param array $style
     */
    public function setStyle($style = null)
    {
        if (!is_null($style) && is_array($style)) {
            $this->style->setStyleByArray($style);
        }
    }

    /**
     * Get section style
     *
     * @return \PhpOffice\PhpWord\Style\Section
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Add header
     *
     * @since 0.10.0
     *
     * @param string $type
     *
     * @return Header
     */
    public function addHeader($type = Header::AUTO)
    {
        return $this->addHeaderFooter($type, true);
    }

    /**
     * Add footer
     *
     * @since 0.10.0
     *
     * @param string $type
     *
     * @return Footer
     */
    public function addFooter($type = Header::AUTO)
    {
        return $this->addHeaderFooter($type, false);
    }

    /**
     * Get header elements
     *
     * @return Header[]
     */
    public function getHeaders()
    {
        return $this->headers;
    }

    /**
     * Get footer elements
     *
     * @return Footer[]
     */
    public function getFooters()
    {
        return $this->footers;
    }

    /**
     * Get the footnote properties
     *
     * @return FootnoteProperties
     */
    public function getFootnoteProperties()
    {
        return $this->footnoteProperties;
    }

    /**
     * Get the footnote properties
     *
     * @deprecated Use the `getFootnoteProperties` method instead
     *
     * @return FootnoteProperties
     *
     * @codeCoverageIgnore
     */
    public function getFootnotePropoperties()
    {
        return $this->footnoteProperties;
    }

    /**
     * Set the footnote properties
     *
     * @param FootnoteProperties $footnoteProperties
     */
    public function setFootnoteProperties(FootnoteProperties $footnoteProperties = null)
    {
        $this->footnoteProperties = $footnoteProperties;
    }

    /**
     * Is there a header for this section that is for the first page only?
     *
     * If any of the Header instances have a type of Header::FIRST then this method returns true.
     * False otherwise.
     *
     * @return bool
     */
    public function hasDifferentFirstPage()
    {
        foreach ($this->headers as $header) {
            if ($header->getType() == Header::FIRST) {
                return true;
            }
        }
        foreach ($this->footers as $footer) {
            if ($footer->getType() == Header::FIRST) {
                return true;
            }
        }

        return false;
    }

    /**
     * Add header/footer
     *
     * @since 0.10.0
     *
     * @param string $type
     * @param bool $header
     *
     * @throws \Exception
     *
     * @return Header|Footer
     */
    private function addHeaderFooter($type = Header::AUTO, $header = true)
    {
        $containerClass = substr(get_class($this), 0, strrpos(get_class($this), '\\')) . '\\' .
            ($header ? 'Header' : 'Footer');
        $collectionArray = $header ? 'headers' : 'footers';
        $collection = &$this->$collectionArray;

        if (in_array($type, array(Header::AUTO, Header::FIRST, Header::EVEN))) {
            $index = count($collection);
            /** @var \PhpOffice\PhpWord\Element\AbstractContainer $container Type hint */
            $container = new $containerClass($this->sectionId, ++$index, $type);
            $container->setPhpWord($this->phpWord);

            $collection[$index] = $container;

            return $container;
        }
        throw new \Exception('Invalid header/footer type.');
    }

    /**
     * Set section style
     *
     * @deprecated 0.12.0
     *
     * @param array $settings
     *
     * @codeCoverageIgnore
     */
    public function setSettings($settings = null)
    {
        $this->setStyle($settings);
    }

    /**
     * Get section style
     *
     * @deprecated 0.12.0
     *
     * @return \PhpOffice\PhpWord\Style\Section
     *
     * @codeCoverageIgnore
     */
    public function getSettings()
    {
        return $this->getStyle();
    }

    /**
     * Create header
     *
     * @deprecated 0.10.0
     *
     * @return Header
     *
     * @codeCoverageIgnore
     */
    public function createHeader()
    {
        return $this->addHeader();
    }

    /**
     * Create footer
     *
     * @deprecated 0.10.0
     *
     * @return Footer
     *
     * @codeCoverageIgnore
     */
    public function createFooter()
    {
        return $this->addFooter();
    }

    /**
     * Get footer
     *
     * @deprecated 0.10.0
     *
     * @return Footer
     *
     * @codeCoverageIgnore
     */
    public function getFooter()
    {
        if (empty($this->footers)) {
            return null;
        }

        return $this->footers[1];
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Shape.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Shape as ShapeStyle;

/**
 * Shape element
 *
 * @since 0.12.0
 */
class Shape extends AbstractElement
{
    /**
     * Shape type arc|curve|line|polyline|rect|oval
     *
     * @var string
     */
    private $type;

    /**
     * Shape style
     *
     * @var \PhpOffice\PhpWord\Style\Shape
     */
    private $style;

    /**
     * Create new instance
     *
     * @param string $type
     * @param mixed $style
     */
    public function __construct($type, $style = null)
    {
        $this->setType($type);
        $this->style = $this->setNewStyle(new ShapeStyle(), $style);
    }

    /**
     * Get type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setType($value = null)
    {
        $enum = array('arc', 'curve', 'line', 'polyline', 'rect', 'oval');
        $this->type = $this->setEnumVal($value, $enum, null);

        return $this;
    }

    /**
     * Get shape style
     *
     * @return \PhpOffice\PhpWord\Style\Shape
     */
    public function getStyle()
    {
        return $this->style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Table as TableStyle;

/**
 * Table element
 */
class Table extends AbstractElement
{
    /**
     * Table style
     *
     * @var \PhpOffice\PhpWord\Style\Table
     */
    private $style;

    /**
     * Table rows
     *
     * @var \PhpOffice\PhpWord\Element\Row[]
     */
    private $rows = array();

    /**
     * Table width
     *
     * @var int
     */
    private $width = null;

    /**
     * Create a new table
     *
     * @param mixed $style
     */
    public function __construct($style = null)
    {
        $this->style = $this->setNewStyle(new TableStyle(), $style);
    }

    /**
     * Add a row
     *
     * @param int $height
     * @param mixed $style
     * @return \PhpOffice\PhpWord\Element\Row
     */
    public function addRow($height = null, $style = null)
    {
        $row = new Row($height, $style);
        $row->setParentContainer($this);
        $this->rows[] = $row;

        return $row;
    }

    /**
     * Add a cell
     *
     * @param int $width
     * @param mixed $style
     * @return \PhpOffice\PhpWord\Element\Cell
     */
    public function addCell($width = null, $style = null)
    {
        $index = count($this->rows) - 1;
        $row = $this->rows[$index];
        $cell = $row->addCell($width, $style);

        return $cell;
    }

    /**
     * Get all rows
     *
     * @return \PhpOffice\PhpWord\Element\Row[]
     */
    public function getRows()
    {
        return $this->rows;
    }

    /**
     * Get table style
     *
     * @return \PhpOffice\PhpWord\Style\Table
     */
    public function getStyle()
    {
        return $this->style;
    }

    /**
     * Get table width
     *
     * @return int
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Set table width.
     *
     * @param int $width
     */
    public function setWidth($width)
    {
        $this->width = $width;
    }

    /**
     * Get column count
     *
     * @return int
     */
    public function countColumns()
    {
        $columnCount = 0;

        $rowCount = count($this->rows);
        for ($i = 0; $i < $rowCount; $i++) {
            /** @var \PhpOffice\PhpWord\Element\Row $row Type hint */
            $row = $this->rows[$i];
            $cellCount = count($row->getCells());
            if ($columnCount < $cellCount) {
                $columnCount = $cellCount;
            }
        }

        return $columnCount;
    }

    /**
     * The first declared cell width for each column
     *
     * @return int[]
     */
    public function findFirstDefinedCellWidths()
    {
        $cellWidths = array();

        foreach ($this->rows as $row) {
            $cells = $row->getCells();
            if (count($cells) <= count($cellWidths)) {
                continue;
            }
            $cellWidths = array();
            foreach ($cells as $cell) {
                $cellWidths[] = $cell->getWidth();
            }
        }

        return $cellWidths;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;

/**
 * Text element
 */
class Text extends AbstractElement
{
    /**
     * Text content
     *
     * @var string
     */
    protected $text;

    /**
     * Text style
     *
     * @var string|\PhpOffice\PhpWord\Style\Font
     */
    protected $fontStyle;

    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    protected $paragraphStyle;

    /**
     * Create a new Text Element
     *
     * @param string $text
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($text = null, $fontStyle = null, $paragraphStyle = null)
    {
        $this->setText($text);
        $paragraphStyle = $this->setParagraphStyle($paragraphStyle);
        $this->setFontStyle($fontStyle, $paragraphStyle);
    }

    /**
     * Set Text style
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Font $style
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $paragraphStyle
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function setFontStyle($style = null, $paragraphStyle = null)
    {
        if ($style instanceof Font) {
            $this->fontStyle = $style;
            $this->setParagraphStyle($paragraphStyle);
        } elseif (is_array($style)) {
            $this->fontStyle = new Font('text', $paragraphStyle);
            $this->fontStyle->setStyleByArray($style);
        } elseif (null === $style) {
            $this->fontStyle = new Font('text', $paragraphStyle);
        } else {
            $this->fontStyle = $style;
            $this->setParagraphStyle($paragraphStyle);
        }

        return $this->fontStyle;
    }

    /**
     * Get Text style
     *
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function getFontStyle()
    {
        return $this->fontStyle;
    }

    /**
     * Set Paragraph style
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $style
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function setParagraphStyle($style = null)
    {
        if (is_array($style)) {
            $this->paragraphStyle = new Paragraph();
            $this->paragraphStyle->setStyleByArray($style);
        } elseif ($style instanceof Paragraph) {
            $this->paragraphStyle = $style;
        } elseif (null === $style) {
            $this->paragraphStyle = new Paragraph();
        } else {
            $this->paragraphStyle = $style;
        }

        return $this->paragraphStyle;
    }

    /**
     * Get Paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Set text content
     *
     * @param string $text
     * @return self
     */
    public function setText($text)
    {
        $this->text = SharedText::toUTF8($text);

        return $this;
    }

    /**
     * Get Text content
     *
     * @return string
     */
    public function getText()
    {
        return $this->text;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/TextBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\TextBox as TextBoxStyle;

/**
 * TextBox element
 *
 * @since 0.11.0
 */
class TextBox extends AbstractContainer
{
    /**
     * @var string Container type
     */
    protected $container = 'TextBox';

    /**
     * TextBox style
     *
     * @var \PhpOffice\PhpWord\Style\TextBox
     */
    private $style;

    /**
     * Create a new textbox
     *
     * @param mixed $style
     */
    public function __construct($style = null)
    {
        $this->style = $this->setNewStyle(new TextBoxStyle(), $style);
    }

    /**
     * Get textbox style
     *
     * @return \PhpOffice\PhpWord\Style\TextBox
     */
    public function getStyle()
    {
        return $this->style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/TextBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;

/**
 * Text break element
 */
class TextBreak extends AbstractElement
{
    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    private $paragraphStyle = null;

    /**
     * Text style
     *
     * @var string|\PhpOffice\PhpWord\Style\Font
     */
    private $fontStyle = null;

    /**
     * Create a new TextBreak Element
     *
     * @param mixed $fontStyle
     * @param mixed $paragraphStyle
     */
    public function __construct($fontStyle = null, $paragraphStyle = null)
    {
        if (!is_null($paragraphStyle)) {
            $paragraphStyle = $this->setParagraphStyle($paragraphStyle);
        }
        if (!is_null($fontStyle)) {
            $this->setFontStyle($fontStyle, $paragraphStyle);
        }
    }

    /**
     * Set Text style
     *
     * @param mixed $style
     * @param mixed $paragraphStyle
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function setFontStyle($style = null, $paragraphStyle = null)
    {
        if ($style instanceof Font) {
            $this->fontStyle = $style;
            $this->setParagraphStyle($paragraphStyle);
        } elseif (is_array($style)) {
            $this->fontStyle = new Font('text', $paragraphStyle);
            $this->fontStyle->setStyleByArray($style);
        } else {
            $this->fontStyle = $style;
            $this->setParagraphStyle($paragraphStyle);
        }

        return $this->fontStyle;
    }

    /**
     * Get Text style
     *
     * @return string|\PhpOffice\PhpWord\Style\Font
     */
    public function getFontStyle()
    {
        return $this->fontStyle;
    }

    /**
     * Set Paragraph style
     *
     * @param   string|array|\PhpOffice\PhpWord\Style\Paragraph $style
     * @return  string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function setParagraphStyle($style = null)
    {
        if (is_array($style)) {
            $this->paragraphStyle = new Paragraph();
            $this->paragraphStyle->setStyleByArray($style);
        } elseif ($style instanceof Paragraph) {
            $this->paragraphStyle = $style;
        } else {
            $this->paragraphStyle = $style;
        }

        return $this->paragraphStyle;
    }

    /**
     * Get Paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Has font/paragraph style defined
     *
     * @return bool
     */
    public function hasStyle()
    {
        return !is_null($this->fontStyle) || !is_null($this->paragraphStyle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/TextRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Style\Paragraph;

/**
 * Textrun/paragraph element
 */
class TextRun extends AbstractContainer
{
    /**
     * @var string Container type
     */
    protected $container = 'TextRun';

    /**
     * Paragraph style
     *
     * @var string|\PhpOffice\PhpWord\Style\Paragraph
     */
    protected $paragraphStyle;

    /**
     * Create new instance
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $paragraphStyle
     */
    public function __construct($paragraphStyle = null)
    {
        $this->paragraphStyle = $this->setParagraphStyle($paragraphStyle);
    }

    /**
     * Get Paragraph style
     *
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraphStyle()
    {
        return $this->paragraphStyle;
    }

    /**
     * Set Paragraph style
     *
     * @param string|array|\PhpOffice\PhpWord\Style\Paragraph $style
     * @return string|\PhpOffice\PhpWord\Style\Paragraph
     */
    public function setParagraphStyle($style = null)
    {
        if (is_array($style)) {
            $this->paragraphStyle = new Paragraph();
            $this->paragraphStyle->setStyleByArray($style);
        } elseif ($style instanceof Paragraph) {
            $this->paragraphStyle = $style;
        } elseif (null === $style) {
            $this->paragraphStyle = new Paragraph();
        } else {
            $this->paragraphStyle = $style;
        }

        return $this->paragraphStyle;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/Title.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style;

/**
 * Title element
 */
class Title extends AbstractElement
{
    /**
     * Title Text content
     *
     * @var string|TextRun
     */
    private $text;

    /**
     * Title depth
     *
     * @var int
     */
    private $depth = 1;

    /**
     * Name of the heading style, e.g. 'Heading1'
     *
     * @var string
     */
    private $style;

    /**
     * Is part of collection
     *
     * @var bool
     */
    protected $collectionRelation = true;

    /**
     * Create a new Title Element
     *
     * @param string|TextRun $text
     * @param int $depth
     */
    public function __construct($text, $depth = 1)
    {
        if (is_string($text)) {
            $this->text = SharedText::toUTF8($text);
        } elseif ($text instanceof TextRun) {
            $this->text = $text;
        } else {
            throw new \InvalidArgumentException('Invalid text, should be a string or a TextRun');
        }

        $this->depth = $depth;
        $styleName = $depth === 0 ? 'Title' : "Heading_{$this->depth}";
        if (array_key_exists($styleName, Style::getStyles())) {
            $this->style = str_replace('_', '', $styleName);
        }
    }

    /**
     * Get Title Text content
     *
     * @return string
     */
    public function getText()
    {
        return $this->text;
    }

    /**
     * Get depth
     *
     * @return int
     */
    public function getDepth()
    {
        return $this->depth;
    }

    /**
     * Get Title style
     *
     * @return string
     */
    public function getStyle()
    {
        return $this->style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/TOC.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\TOC as TOCStyle;

/**
 * Table of contents
 */
class TOC extends AbstractElement
{
    /**
     * TOC style
     *
     * @var \PhpOffice\PhpWord\Style\TOC
     */
    private $TOCStyle;

    /**
     * Font style
     *
     * @var \PhpOffice\PhpWord\Style\Font|string
     */
    private $fontStyle;

    /**
     * Min title depth to show
     *
     * @var int
     */
    private $minDepth = 1;

    /**
     * Max title depth to show
     *
     * @var int
     */
    private $maxDepth = 9;

    /**
     * Create a new Table-of-Contents Element
     *
     * @param mixed $fontStyle
     * @param array $tocStyle
     * @param int $minDepth
     * @param int $maxDepth
     */
    public function __construct($fontStyle = null, $tocStyle = null, $minDepth = 1, $maxDepth = 9)
    {
        $this->TOCStyle = new TOCStyle();

        if (!is_null($tocStyle) && is_array($tocStyle)) {
            $this->TOCStyle->setStyleByArray($tocStyle);
        }

        if (!is_null($fontStyle) && is_array($fontStyle)) {
            $this->fontStyle = new Font();
            $this->fontStyle->setStyleByArray($fontStyle);
        } else {
            $this->fontStyle = $fontStyle;
        }

        $this->minDepth = $minDepth;
        $this->maxDepth = $maxDepth;
    }

    /**
     * Get all titles
     *
     * @return array
     */
    public function getTitles()
    {
        if (!$this->phpWord instanceof PhpWord) {
            return array();
        }

        $titles = $this->phpWord->getTitles()->getItems();
        foreach ($titles as $i => $title) {
            /** @var \PhpOffice\PhpWord\Element\Title $title Type hint */
            $depth = $title->getDepth();
            if ($this->minDepth > $depth) {
                unset($titles[$i]);
            }
            if (($this->maxDepth != 0) && ($this->maxDepth < $depth)) {
                unset($titles[$i]);
            }
        }

        return $titles;
    }

    /**
     * Get TOC Style
     *
     * @return \PhpOffice\PhpWord\Style\TOC
     */
    public function getStyleTOC()
    {
        return $this->TOCStyle;
    }

    /**
     * Get Font Style
     *
     * @return \PhpOffice\PhpWord\Style\Font|string
     */
    public function getStyleFont()
    {
        return $this->fontStyle;
    }

    /**
     * Set max depth.
     *
     * @param int $value
     */
    public function setMaxDepth($value)
    {
        $this->maxDepth = $value;
    }

    /**
     * Get Max Depth
     *
     * @return int Max depth of titles
     */
    public function getMaxDepth()
    {
        return $this->maxDepth;
    }

    /**
     * Set min depth.
     *
     * @param int $value
     */
    public function setMinDepth($value)
    {
        $this->minDepth = $value;
    }

    /**
     * Get Min Depth
     *
     * @return int Min depth of titles
     */
    public function getMinDepth()
    {
        return $this->minDepth;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Element/TrackChange.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Element;

/**
 * TrackChange element
 * @see http://datypic.com/sc/ooxml/t-w_CT_TrackChange.html
 * @see http://datypic.com/sc/ooxml/t-w_CT_RunTrackChange.html
 */
class TrackChange extends AbstractContainer
{
    const INSERTED = 'INSERTED';
    const DELETED = 'DELETED';

    /**
     * @var string Container type
     */
    protected $container = 'TrackChange';

    /**
     * The type of change, (insert or delete), not applicable for PhpOffice\PhpWord\Element\Comment
     *
     * @var string
     */
    private $changeType;

    /**
     * Author
     *
     * @var string
     */
    private $author;

    /**
     * Date
     *
     * @var \DateTime
     */
    private $date;

    /**
     * Create a new TrackChange Element
     *
     * @param string $changeType
     * @param string $author
     * @param null|int|bool|\DateTime $date
     */
    public function __construct($changeType = null, $author = null, $date = null)
    {
        $this->changeType = $changeType;
        $this->author = $author;
        if ($date !== null && $date !== false) {
            $this->date = ($date instanceof \DateTime) ? $date : new \DateTime('@' . $date);
        }
    }

    /**
     * Get TrackChange Author
     *
     * @return string
     */
    public function getAuthor()
    {
        return $this->author;
    }

    /**
     * Get TrackChange Date
     *
     * @return \DateTime
     */
    public function getDate()
    {
        return $this->date;
    }

    /**
     * Get the Change type
     *
     * @return string
     */
    public function getChangeType()
    {
        return $this->changeType;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Escaper/AbstractEscaper.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Escaper;

/**
 * @since 0.13.0
 *
 * @codeCoverageIgnore
 */
abstract class AbstractEscaper implements EscaperInterface
{
    /**
     * @param string $input
     *
     * @return string
     */
    abstract protected function escapeSingleValue($input);

    public function escape($input)
    {
        if (is_array($input)) {
            foreach ($input as &$item) {
                $item = $this->escapeSingleValue($item);
            }
        } else {
            $input = $this->escapeSingleValue($input);
        }

        return $input;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Escaper/EscaperInterface.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Escaper;

/**
 * @since 0.13.0
 *
 * @codeCoverageIgnore
 */
interface EscaperInterface
{
    /**
     * @param mixed $input
     *
     * @return mixed
     */
    public function escape($input);
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Escaper/RegExp.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Escaper;

/**
 * @since 0.13.0
 *
 * @codeCoverageIgnore
 */
class RegExp extends AbstractEscaper
{
    const REG_EXP_DELIMITER = '/';

    protected function escapeSingleValue($input)
    {
        return self::REG_EXP_DELIMITER . preg_quote($input, self::REG_EXP_DELIMITER) . self::REG_EXP_DELIMITER . 'u';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Escaper/Rtf.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Escaper;

/**
 * @since 0.13.0
 *
 * @codeCoverageIgnore
 */
class Rtf extends AbstractEscaper
{
    protected function escapeAsciiCharacter($code)
    {
        if ($code == 9) {
            return '{\\tab}';
        }
        if (0x20 > $code || $code >= 0x80) {
            return '{\\u' . $code . '}';
        }
        if ($code == 123 || $code == 125 || $code == 92) { // open or close brace or backslash
            return '\\' . chr($code);
        }

        return chr($code);
    }

    protected function escapeMultibyteCharacter($code)
    {
        return '\\uc0{\\u' . $code . '}';
    }

    /**
     * @see http://www.randomchaos.com/documents/?source=php_and_unicode
     * @param string $input
     */
    protected function escapeSingleValue($input)
    {
        $escapedValue = '';

        $numberOfBytes = 1;
        $bytes = array();
        for ($i = 0; $i < strlen($input); ++$i) {
            $character = $input[$i];
            $asciiCode = ord($character);

            if ($asciiCode < 128) {
                $escapedValue .= $this->escapeAsciiCharacter($asciiCode);
            } else {
                if (0 == count($bytes)) {
                    if ($asciiCode < 224) {
                        $numberOfBytes = 2;
                    } elseif ($asciiCode < 240) {
                        $numberOfBytes = 3;
                    } elseif ($asciiCode < 248) {
                        $numberOfBytes = 4;
                    }
                }

                $bytes[] = $asciiCode;

                if ($numberOfBytes == count($bytes)) {
                    if (4 == $numberOfBytes) {
                        $multibyteCode = ($bytes[0] % 8) * 262144 + ($bytes[1] % 64) * 4096 + ($bytes[2] % 64) * 64 + ($bytes[3] % 64);
                    } elseif (3 == $numberOfBytes) {
                        $multibyteCode = ($bytes[0] % 16) * 4096 + ($bytes[1] % 64) * 64 + ($bytes[2] % 64);
                    } else {
                        $multibyteCode = ($bytes[0] % 32) * 64 + ($bytes[1] % 64);
                    }

                    if (65279 != $multibyteCode) {
                        $escapedValue .= $multibyteCode < 128 ? $this->escapeAsciiCharacter($multibyteCode) : $this->escapeMultibyteCharacter($multibyteCode);
                    }

                    $numberOfBytes = 1;
                    $bytes = array();
                }
            }
        }

        return $escapedValue;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Escaper/Xml.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Escaper;

/**
 * @since 0.13.0
 *
 * @codeCoverageIgnore
 */
class Xml extends AbstractEscaper
{
    protected function escapeSingleValue($input)
    {
        // todo: omit encoding parameter after migration onto PHP 5.4
        return htmlspecialchars($input, ENT_QUOTES, 'UTF-8');
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/CopyFileException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * @since 0.12.0
 */
final class CopyFileException extends Exception
{
    /**
     * @param string $source The fully qualified source file name
     * @param string $destination The fully qualified destination file name
     * @param int $code The user defined exception code
     * @param \Exception $previous The previous exception used for the exception chaining
     */
    final public function __construct($source, $destination, $code = 0, \Exception $previous = null)
    {
        parent::__construct(
            sprintf('Could not copy \'%s\' file to \'%s\'.', $source, $destination),
            $code,
            $previous
        );
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/CreateTemporaryFileException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * @since 0.12.0
 */
final class CreateTemporaryFileException extends Exception
{
    /**
     * @param int $code The user defined exception code
     * @param \Exception $previous The previous exception used for the exception chaining
     */
    final public function __construct($code = 0, \Exception $previous = null)
    {
        parent::__construct(
            'Could not create a temporary file with unique name in the specified directory.',
            $code,
            $previous
        );
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/Exception.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * General exception
 */
class Exception extends \Exception
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/InvalidImageException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * Exception used for when an image is not found
 */
class InvalidImageException extends Exception
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/InvalidObjectException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * Exception used for when an image is not found
 */
class InvalidObjectException extends Exception
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/InvalidStyleException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

use InvalidArgumentException;

/**
 * Exception used for when a style value is invalid
 */
class InvalidStyleException extends InvalidArgumentException
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Exception/UnsupportedImageTypeException.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Exception;

/**
 * Exception used for when an image type is unsupported
 */
class UnsupportedImageTypeException extends Exception
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/IOFactory.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Reader\ReaderInterface;
use PhpOffice\PhpWord\Writer\WriterInterface;

abstract class IOFactory
{
    /**
     * Create new writer
     *
     * @param PhpWord $phpWord
     * @param string $name
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return WriterInterface
     */
    public static function createWriter(PhpWord $phpWord, $name = 'Word2007')
    {
        if ($name !== 'WriterInterface' && !in_array($name, array('ODText', 'RTF', 'Word2007', 'HTML', 'PDF'), true)) {
            throw new Exception("\"{$name}\" is not a valid writer.");
        }

        $fqName = "PhpOffice\\PhpWord\\Writer\\{$name}";

        return new $fqName($phpWord);
    }

    /**
     * Create new reader
     *
     * @param string $name
     *
     * @throws Exception
     *
     * @return ReaderInterface
     */
    public static function createReader($name = 'Word2007')
    {
        return self::createObject('Reader', $name);
    }

    /**
     * Create new object
     *
     * @param string $type
     * @param string $name
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return \PhpOffice\PhpWord\Writer\WriterInterface|\PhpOffice\PhpWord\Reader\ReaderInterface
     */
    private static function createObject($type, $name, $phpWord = null)
    {
        $class = "PhpOffice\\PhpWord\\{$type}\\{$name}";
        if (class_exists($class) && self::isConcreteClass($class)) {
            return new $class($phpWord);
        }
        throw new Exception("\"{$name}\" is not a valid {$type}.");
    }

    /**
     * Loads PhpWord from file
     *
     * @param string $filename The name of the file
     * @param string $readerName
     * @return \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public static function load($filename, $readerName = 'Word2007')
    {
        /** @var \PhpOffice\PhpWord\Reader\ReaderInterface $reader */
        $reader = self::createReader($readerName);

        return $reader->load($filename);
    }

    /**
     * Check if it's a concrete class (not abstract nor interface)
     *
     * @param string $class
     * @return bool
     */
    private static function isConcreteClass($class)
    {
        $reflection = new \ReflectionClass($class);

        return !$reflection->isAbstract() && !$reflection->isInterface();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Media.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

use PhpOffice\PhpWord\Element\Image;
use PhpOffice\PhpWord\Exception\Exception;

/**
 * Media collection
 */
class Media
{
    /**
     * Media elements
     *
     * @var array
     */
    private static $elements = array();

    /**
     * Add new media element
     *
     * @since 0.10.0
     * @since 0.9.2
     *
     * @param string $container section|headerx|footerx|footnote|endnote
     * @param string $mediaType image|object|link
     * @param string $source
     * @param \PhpOffice\PhpWord\Element\Image $image
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return int
     */
    public static function addElement($container, $mediaType, $source, Image $image = null)
    {
        // Assign unique media Id and initiate media container if none exists
        $mediaId = md5($container . $source);
        if (!isset(self::$elements[$container])) {
            self::$elements[$container] = array();
        }

        // Add media if not exists or point to existing media
        if (!isset(self::$elements[$container][$mediaId])) {
            $mediaCount = self::countElements($container);
            $mediaTypeCount = self::countElements($container, $mediaType);
            $mediaTypeCount++;
            $rId = ++$mediaCount;
            $target = null;
            $mediaData = array('mediaIndex' => $mediaTypeCount);

            switch ($mediaType) {
                // Images
                case 'image':
                    if (is_null($image)) {
                        throw new Exception('Image object not assigned.');
                    }
                    $isMemImage = $image->isMemImage();
                    $extension = $image->getImageExtension();
                    $mediaData['imageExtension'] = $extension;
                    $mediaData['imageType'] = $image->getImageType();
                    if ($isMemImage) {
                        $mediaData['isMemImage'] = true;
                        $mediaData['createFunction'] = $image->getImageCreateFunction();
                        $mediaData['imageFunction'] = $image->getImageFunction();
                    }
                    $target = "{$container}_image{$mediaTypeCount}.{$extension}";
                    $image->setTarget($target);
                    $image->setMediaIndex($mediaTypeCount);
                    break;
                // Objects
                case 'object':
                    $target = "{$container}_oleObject{$mediaTypeCount}.bin";
                    break;
                // Links
                case 'link':
                    $target = $source;
                    break;
            }

            $mediaData['source'] = $source;
            $mediaData['target'] = $target;
            $mediaData['type'] = $mediaType;
            $mediaData['rID'] = $rId;
            self::$elements[$container][$mediaId] = $mediaData;

            return $rId;
        }

        $mediaData = self::$elements[$container][$mediaId];
        if (!is_null($image)) {
            $image->setTarget($mediaData['target']);
            $image->setMediaIndex($mediaData['mediaIndex']);
        }

        return $mediaData['rID'];
    }

    /**
     * Get media elements count
     *
     * @param string $container section|headerx|footerx|footnote|endnote
     * @param string $mediaType image|object|link
     * @return int
     * @since 0.10.0
     */
    public static function countElements($container, $mediaType = null)
    {
        $mediaCount = 0;

        if (isset(self::$elements[$container])) {
            foreach (self::$elements[$container] as $mediaData) {
                if (!is_null($mediaType)) {
                    if ($mediaType == $mediaData['type']) {
                        $mediaCount++;
                    }
                } else {
                    $mediaCount++;
                }
            }
        }

        return $mediaCount;
    }

    /**
     * Get media elements
     *
     * @param string $container section|headerx|footerx|footnote|endnote
     * @param string $type image|object|link
     * @return array
     * @since 0.10.0
     */
    public static function getElements($container, $type = null)
    {
        $elements = array();

        // If header/footer, search for headerx and footerx where x is number
        if ($container == 'header' || $container == 'footer') {
            foreach (self::$elements as $key => $val) {
                if (substr($key, 0, 6) == $container) {
                    $elements[$key] = $val;
                }
            }

            return $elements;
        }

        if (!isset(self::$elements[$container])) {
            return $elements;
        }

        return self::getElementsByType($container, $type);
    }

    /**
     * Get elements by media type
     *
     * @param string $container section|footnote|endnote
     * @param string $type image|object|link
     * @return array
     * @since 0.11.0 Splitted from `getElements` to reduce complexity
     */
    private static function getElementsByType($container, $type = null)
    {
        $elements = array();

        foreach (self::$elements[$container] as $key => $data) {
            if ($type !== null) {
                if ($type == $data['type']) {
                    $elements[$key] = $data;
                }
            } else {
                $elements[$key] = $data;
            }
        }

        return $elements;
    }

    /**
     * Reset media elements
     */
    public static function resetElements()
    {
        self::$elements = array();
    }

    /**
     * Add new Section Media Element
     *
     * @deprecated 0.10.0
     *
     * @param  string $src
     * @param  string $type
     * @param  \PhpOffice\PhpWord\Element\Image $image
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function addSectionMediaElement($src, $type, Image $image = null)
    {
        return self::addElement('section', $type, $src, $image);
    }

    /**
     * Add new Section Link Element
     *
     * @deprecated 0.10.0
     *
     * @param string $linkSrc
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function addSectionLinkElement($linkSrc)
    {
        return self::addElement('section', 'link', $linkSrc);
    }

    /**
     * Get Section Media Elements
     *
     * @deprecated 0.10.0
     *
     * @param string $key
     *
     * @return array
     *
     * @codeCoverageIgnore
     */
    public static function getSectionMediaElements($key = null)
    {
        return self::getElements('section', $key);
    }

    /**
     * Get Section Media Elements Count
     *
     * @deprecated 0.10.0
     *
     * @param string $key
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function countSectionMediaElements($key = null)
    {
        return self::countElements('section', $key);
    }

    /**
     * Add new Header Media Element
     *
     * @deprecated 0.10.0
     *
     * @param  int $headerCount
     * @param  string $src
     * @param  \PhpOffice\PhpWord\Element\Image $image
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function addHeaderMediaElement($headerCount, $src, Image $image = null)
    {
        return self::addElement("header{$headerCount}", 'image', $src, $image);
    }

    /**
     * Get Header Media Elements Count
     *
     * @deprecated 0.10.0
     *
     * @param string $key
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function countHeaderMediaElements($key)
    {
        return self::countElements($key);
    }

    /**
     * Get Header Media Elements
     *
     * @deprecated 0.10.0
     *
     * @return array
     *
     * @codeCoverageIgnore
     */
    public static function getHeaderMediaElements()
    {
        return self::getElements('header');
    }

    /**
     * Add new Footer Media Element
     *
     * @deprecated 0.10.0
     *
     * @param  int $footerCount
     * @param  string $src
     * @param  \PhpOffice\PhpWord\Element\Image $image
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function addFooterMediaElement($footerCount, $src, Image $image = null)
    {
        return self::addElement("footer{$footerCount}", 'image', $src, $image);
    }

    /**
     * Get Footer Media Elements Count
     *
     * @deprecated 0.10.0
     *
     * @param string $key
     *
     * @return int
     *
     * @codeCoverageIgnore
     */
    public static function countFooterMediaElements($key)
    {
        return self::countElements($key);
    }

    /**
     * Get Footer Media Elements
     *
     * @deprecated 0.10.0
     *
     * @return array
     *
     * @codeCoverageIgnore
     */
    public static function getFooterMediaElements()
    {
        return self::getElements('footer');
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Metadata/Compatibility.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Metadata;

/**
 * Compatibility setting class
 *
 * @since 0.12.0
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_Compat.html
 */
class Compatibility
{
    /**
     * OOXML version
     *
     * 12 = 2007
     * 14 = 2010
     * 15 = 2013
     *
     * @var int
     * @see  http://msdn.microsoft.com/en-us/library/dd909048%28v=office.12%29.aspx
     */
    private $ooxmlVersion = 12;

    /**
     * Get OOXML version
     *
     * @return int
     */
    public function getOoxmlVersion()
    {
        return $this->ooxmlVersion;
    }

    /**
     * Set OOXML version
     *
     * @param int $value
     * @return self
     */
    public function setOoxmlVersion($value)
    {
        $this->ooxmlVersion = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Metadata/DocInfo.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Metadata;

/**
 * Document information
 */
class DocInfo
{
    /** @const string Property type constants */
    const PROPERTY_TYPE_BOOLEAN = 'b';
    const PROPERTY_TYPE_INTEGER = 'i';
    const PROPERTY_TYPE_FLOAT = 'f';
    const PROPERTY_TYPE_DATE = 'd';
    const PROPERTY_TYPE_STRING = 's';
    const PROPERTY_TYPE_UNKNOWN = 'u';

    /**
     * Creator
     *
     * @var string
     */
    private $creator;

    /**
     * LastModifiedBy
     *
     * @var string
     */
    private $lastModifiedBy;

    /**
     * Created
     *
     * @var int
     */
    private $created;

    /**
     * Modified
     *
     * @var int
     */
    private $modified;

    /**
     * Title
     *
     * @var string
     */
    private $title;

    /**
     * Description
     *
     * @var string
     */
    private $description;

    /**
     * Subject
     *
     * @var string
     */
    private $subject;

    /**
     * Keywords
     *
     * @var string
     */
    private $keywords;

    /**
     * Category
     *
     * @var string
     */
    private $category;

    /**
     * Company
     *
     * @var string
     */
    private $company;

    /**
     * Manager
     *
     * @var string
     */
    private $manager;

    /**
     * Custom Properties
     *
     * @var array
     */
    private $customProperties = array();

    /**
     * Create new instance
     */
    public function __construct()
    {
        $this->creator = '';
        $this->lastModifiedBy = $this->creator;
        $this->created = time();
        $this->modified = time();
        $this->title = '';
        $this->subject = '';
        $this->description = '';
        $this->keywords = '';
        $this->category = '';
        $this->company = '';
        $this->manager = '';
    }

    /**
     * Get Creator
     *
     * @return string
     */
    public function getCreator()
    {
        return $this->creator;
    }

    /**
     * Set Creator
     *
     * @param  string $value
     * @return self
     */
    public function setCreator($value = '')
    {
        $this->creator = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Last Modified By
     *
     * @return string
     */
    public function getLastModifiedBy()
    {
        return $this->lastModifiedBy;
    }

    /**
     * Set Last Modified By
     *
     * @param  string $value
     * @return self
     */
    public function setLastModifiedBy($value = '')
    {
        $this->lastModifiedBy = $this->setValue($value, $this->creator);

        return $this;
    }

    /**
     * Get Created
     *
     * @return int
     */
    public function getCreated()
    {
        return $this->created;
    }

    /**
     * Set Created
     *
     * @param  int $value
     * @return self
     */
    public function setCreated($value = null)
    {
        $this->created = $this->setValue($value, time());

        return $this;
    }

    /**
     * Get Modified
     *
     * @return int
     */
    public function getModified()
    {
        return $this->modified;
    }

    /**
     * Set Modified
     *
     * @param  int $value
     * @return self
     */
    public function setModified($value = null)
    {
        $this->modified = $this->setValue($value, time());

        return $this;
    }

    /**
     * Get Title
     *
     * @return string
     */
    public function getTitle()
    {
        return $this->title;
    }

    /**
     * Set Title
     *
     * @param  string $value
     * @return self
     */
    public function setTitle($value = '')
    {
        $this->title = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Description
     *
     * @return string
     */
    public function getDescription()
    {
        return $this->description;
    }

    /**
     * Set Description
     *
     * @param  string $value
     * @return self
     */
    public function setDescription($value = '')
    {
        $this->description = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Subject
     *
     * @return string
     */
    public function getSubject()
    {
        return $this->subject;
    }

    /**
     * Set Subject
     *
     * @param  string $value
     * @return self
     */
    public function setSubject($value = '')
    {
        $this->subject = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Keywords
     *
     * @return string
     */
    public function getKeywords()
    {
        return $this->keywords;
    }

    /**
     * Set Keywords
     *
     * @param string $value
     * @return self
     */
    public function setKeywords($value = '')
    {
        $this->keywords = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Category
     *
     * @return string
     */
    public function getCategory()
    {
        return $this->category;
    }

    /**
     * Set Category
     *
     * @param string $value
     * @return self
     */
    public function setCategory($value = '')
    {
        $this->category = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Company
     *
     * @return string
     */
    public function getCompany()
    {
        return $this->company;
    }

    /**
     * Set Company
     *
     * @param string $value
     * @return self
     */
    public function setCompany($value = '')
    {
        $this->company = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get Manager
     *
     * @return string
     */
    public function getManager()
    {
        return $this->manager;
    }

    /**
     * Set Manager
     *
     * @param string $value
     * @return self
     */
    public function setManager($value = '')
    {
        $this->manager = $this->setValue($value, '');

        return $this;
    }

    /**
     * Get a List of Custom Property Names
     *
     * @return array of string
     */
    public function getCustomProperties()
    {
        return array_keys($this->customProperties);
    }

    /**
     * Check if a Custom Property is defined
     *
     * @param string $propertyName
     * @return bool
     */
    public function isCustomPropertySet($propertyName)
    {
        return isset($this->customProperties[$propertyName]);
    }

    /**
     * Get a Custom Property Value
     *
     * @param string $propertyName
     * @return mixed
     */
    public function getCustomPropertyValue($propertyName)
    {
        if ($this->isCustomPropertySet($propertyName)) {
            return $this->customProperties[$propertyName]['value'];
        }

        return null;
    }

    /**
     * Get a Custom Property Type
     *
     * @param string $propertyName
     * @return string
     */
    public function getCustomPropertyType($propertyName)
    {
        if ($this->isCustomPropertySet($propertyName)) {
            return $this->customProperties[$propertyName]['type'];
        }

        return null;
    }

    /**
     * Set a Custom Property
     *
     * @param string $propertyName
     * @param mixed $propertyValue
     * @param string $propertyType
     *   'i': Integer
     *   'f': Floating Point
     *   's': String
     *   'd': Date/Time
     *   'b': Boolean
     * @return self
     */
    public function setCustomProperty($propertyName, $propertyValue = '', $propertyType = null)
    {
        $propertyTypes = array(
            self::PROPERTY_TYPE_INTEGER,
            self::PROPERTY_TYPE_FLOAT,
            self::PROPERTY_TYPE_STRING,
            self::PROPERTY_TYPE_DATE,
            self::PROPERTY_TYPE_BOOLEAN,
        );
        if (($propertyType === null) || (!in_array($propertyType, $propertyTypes))) {
            if ($propertyValue === null) {
                $propertyType = self::PROPERTY_TYPE_STRING;
            } elseif (is_float($propertyValue)) {
                $propertyType = self::PROPERTY_TYPE_FLOAT;
            } elseif (is_int($propertyValue)) {
                $propertyType = self::PROPERTY_TYPE_INTEGER;
            } elseif (is_bool($propertyValue)) {
                $propertyType = self::PROPERTY_TYPE_BOOLEAN;
            } elseif ($propertyValue instanceof \DateTime) {
                $propertyType = self::PROPERTY_TYPE_DATE;
            } else {
                $propertyType = self::PROPERTY_TYPE_STRING;
            }
        }

        $this->customProperties[$propertyName] = array(
            'value' => $propertyValue,
            'type'  => $propertyType,
        );

        return $this;
    }

    /**
     * Convert document property based on type
     *
     * @param string $propertyValue
     * @param string $propertyType
     * @return mixed
     */
    public static function convertProperty($propertyValue, $propertyType)
    {
        $conversion = self::getConversion($propertyType);

        switch ($conversion) {
            case 'empty': // Empty
                return '';
            case 'null': // Null
                return null;
            case 'int': // Signed integer
                return (int) $propertyValue;
            case 'uint': // Unsigned integer
                return abs((int) $propertyValue);
            case 'float': // Float
                return (float) $propertyValue;
            case 'date': // Date
                return strtotime($propertyValue);
            case 'bool': // Boolean
                return $propertyValue == 'true';
        }

        return $propertyValue;
    }

    /**
     * Convert document property type
     *
     * @param string $propertyType
     * @return string
     */
    public static function convertPropertyType($propertyType)
    {
        $typeGroups = array(
            self::PROPERTY_TYPE_INTEGER => array('i1', 'i2', 'i4', 'i8', 'int', 'ui1', 'ui2', 'ui4', 'ui8', 'uint'),
            self::PROPERTY_TYPE_FLOAT   => array('r4', 'r8', 'decimal'),
            self::PROPERTY_TYPE_STRING  => array('empty', 'null', 'lpstr', 'lpwstr', 'bstr'),
            self::PROPERTY_TYPE_DATE    => array('date', 'filetime'),
            self::PROPERTY_TYPE_BOOLEAN => array('bool'),
        );
        foreach ($typeGroups as $groupId => $groupMembers) {
            if (in_array($propertyType, $groupMembers)) {
                return $groupId;
            }
        }

        return self::PROPERTY_TYPE_UNKNOWN;
    }

    /**
     * Set default for null and empty value
     *
     * @param mixed $value
     * @param mixed $default
     * @return mixed
     */
    private function setValue($value, $default)
    {
        if ($value === null || $value == '') {
            $value = $default;
        }

        return $value;
    }

    /**
     * Get conversion model depending on property type
     *
     * @param string $propertyType
     * @return string
     */
    private static function getConversion($propertyType)
    {
        $conversions = array(
            'empty' => array('empty'),
            'null'  => array('null'),
            'int'   => array('i1', 'i2', 'i4', 'i8', 'int'),
            'uint'  => array('ui1', 'ui2', 'ui4', 'ui8', 'uint'),
            'float' => array('r4', 'r8', 'decimal'),
            'bool'  => array('bool'),
            'date'  => array('date', 'filetime'),
        );
        foreach ($conversions as $conversion => $types) {
            if (in_array($propertyType, $types)) {
                return $conversion;
            }
        }

        return 'string';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Metadata/Protection.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Metadata;

use PhpOffice\PhpWord\Shared\Microsoft\PasswordEncoder;
use PhpOffice\PhpWord\SimpleType\DocProtect;

/**
 * Document protection class
 *
 * @since 0.12.0
 * @see http://www.datypic.com/sc/ooxml/t-w_CT_DocProtect.html
 */
class Protection
{
    /**
     * Editing restriction none|readOnly|comments|trackedChanges|forms
     *
     * @var string
     * @see  http://www.datypic.com/sc/ooxml/a-w_edit-1.html
     */
    private $editing;

    /**
     * password
     *
     * @var string
     */
    private $password;

    /**
     * Iterations to Run Hashing Algorithm
     *
     * @var int
     */
    private $spinCount = 100000;

    /**
     * Cryptographic Hashing Algorithm (see constants defined in \PhpOffice\PhpWord\Shared\Microsoft\PasswordEncoder)
     *
     * @var string
     */
    private $algorithm = PasswordEncoder::ALGORITHM_SHA_1;

    /**
     * Salt for Password Verifier
     *
     * @var string
     */
    private $salt;

    /**
     * Create a new instance
     *
     * @param string $editing
     */
    public function __construct($editing = null)
    {
        if ($editing != null) {
            $this->setEditing($editing);
        }
    }

    /**
     * Get editing protection
     *
     * @return string
     */
    public function getEditing()
    {
        return $this->editing;
    }

    /**
     * Set editing protection
     *
     * @param string $editing Any value of \PhpOffice\PhpWord\SimpleType\DocProtect
     * @return self
     */
    public function setEditing($editing = null)
    {
        DocProtect::validate($editing);
        $this->editing = $editing;

        return $this;
    }

    /**
     * Get password
     *
     * @return string
     */
    public function getPassword()
    {
        return $this->password;
    }

    /**
     * Set password
     *
     * @param string $password
     * @return self
     */
    public function setPassword($password)
    {
        $this->password = $password;

        return $this;
    }

    /**
     * Get count for hash iterations
     *
     * @return int
     */
    public function getSpinCount()
    {
        return $this->spinCount;
    }

    /**
     * Set count for hash iterations
     *
     * @param int $spinCount
     * @return self
     */
    public function setSpinCount($spinCount)
    {
        $this->spinCount = $spinCount;

        return $this;
    }

    /**
     * Get algorithm
     *
     * @return string
     */
    public function getAlgorithm()
    {
        return $this->algorithm;
    }

    /**
     * Set algorithm
     *
     * @param string $algorithm
     * @return self
     */
    public function setAlgorithm($algorithm)
    {
        $this->algorithm = $algorithm;

        return $this;
    }

    /**
     * Get salt
     *
     * @return string
     */
    public function getSalt()
    {
        return $this->salt;
    }

    /**
     * Set salt. Salt HAS to be 16 characters, or an exception will be thrown.
     *
     * @param string $salt
     * @throws \InvalidArgumentException
     * @return self
     */
    public function setSalt($salt)
    {
        if ($salt !== null && strlen($salt) !== 16) {
            throw new \InvalidArgumentException('salt has to be of exactly 16 bytes length');
        }

        $this->salt = $salt;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Metadata/Settings.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Metadata;

use PhpOffice\PhpWord\ComplexType\ProofState;
use PhpOffice\PhpWord\ComplexType\TrackChangesView;
use PhpOffice\PhpWord\SimpleType\Zoom;
use PhpOffice\PhpWord\Style\Language;

/**
 * Setting class
 *
 * @since 0.14.0
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_Settings.html
 */
class Settings
{
    /**
     * Magnification Setting
     *
     * @see  http://www.datypic.com/sc/ooxml/e-w_zoom-1.html
     * @var mixed either integer, in which case it treated as a percent, or one of PhpOffice\PhpWord\SimpleType\Zoom
     */
    private $zoom = 100;

    /**
     * Mirror Page Margins
     *
     * @see http://www.datypic.com/sc/ooxml/e-w_mirrorMargins-1.html
     * @var bool
     */
    private $mirrorMargins;

    /**
     * Hide spelling errors
     *
     * @var bool
     */
    private $hideSpellingErrors = false;

    /**
     * Hide grammatical errors
     *
     * @var bool
     */
    private $hideGrammaticalErrors = false;

    /**
     * Visibility of Annotation Types
     *
     * @var TrackChangesView
     */
    private $revisionView;

    /**
     * Track Revisions to Document
     *
     * @var bool
     */
    private $trackRevisions = false;

    /**
     * Do Not Use Move Syntax When Tracking Revisions
     *
     * @var bool
     */
    private $doNotTrackMoves = false;

    /**
     * Do Not Track Formatting Revisions When Tracking Revisions
     *
     * @var bool
     */
    private $doNotTrackFormatting = false;

    /**
     * Spelling and Grammatical Checking State
     *
     * @var \PhpOffice\PhpWord\ComplexType\ProofState
     */
    private $proofState;

    /**
     * Document Editing Restrictions
     *
     * @var \PhpOffice\PhpWord\Metadata\Protection
     */
    private $documentProtection;

    /**
     * Enables different header for odd and even pages.
     *
     * @var bool
     */
    private $evenAndOddHeaders = false;

    /**
     * Theme Font Languages
     *
     * @var Language
     */
    private $themeFontLang;

    /**
     * Automatically Recalculate Fields on Open
     *
     * @var bool
     */
    private $updateFields = false;

    /**
     * Radix Point for Field Code Evaluation
     *
     * @var string
     */
    private $decimalSymbol = '.';

    /**
     * Automatically hyphenate document contents when displayed
     *
     * @var bool|null
     */
    private $autoHyphenation;

    /**
     * Maximum number of consecutively hyphenated lines
     *
     * @var int|null
     */
    private $consecutiveHyphenLimit;

    /**
     * The allowed amount of whitespace before hyphenation is applied
     * @var float|null
     */
    private $hyphenationZone;

    /**
     * Do not hyphenate words in all capital letters
     * @var bool|null
     */
    private $doNotHyphenateCaps;

    /**
     * @return Protection
     */
    public function getDocumentProtection()
    {
        if ($this->documentProtection == null) {
            $this->documentProtection = new Protection();
        }

        return $this->documentProtection;
    }

    /**
     * @param Protection $documentProtection
     */
    public function setDocumentProtection($documentProtection)
    {
        $this->documentProtection = $documentProtection;
    }

    /**
     * @return ProofState
     */
    public function getProofState()
    {
        if ($this->proofState == null) {
            $this->proofState = new ProofState();
        }

        return $this->proofState;
    }

    /**
     * @param ProofState $proofState
     */
    public function setProofState($proofState)
    {
        $this->proofState = $proofState;
    }

    /**
     * Are spelling errors hidden
     *
     * @return bool
     */
    public function hasHideSpellingErrors()
    {
        return $this->hideSpellingErrors;
    }

    /**
     * Hide spelling errors
     *
     * @param bool $hideSpellingErrors
     */
    public function setHideSpellingErrors($hideSpellingErrors)
    {
        $this->hideSpellingErrors = $hideSpellingErrors === null ? true : $hideSpellingErrors;
    }

    /**
     * Are grammatical errors hidden
     *
     * @return bool
     */
    public function hasHideGrammaticalErrors()
    {
        return $this->hideGrammaticalErrors;
    }

    /**
     * Hide grammatical errors
     *
     * @param bool $hideGrammaticalErrors
     */
    public function setHideGrammaticalErrors($hideGrammaticalErrors)
    {
        $this->hideGrammaticalErrors = $hideGrammaticalErrors === null ? true : $hideGrammaticalErrors;
    }

    /**
     * @return bool
     */
    public function hasEvenAndOddHeaders()
    {
        return $this->evenAndOddHeaders;
    }

    /**
     * @param bool $evenAndOddHeaders
     */
    public function setEvenAndOddHeaders($evenAndOddHeaders)
    {
        $this->evenAndOddHeaders = $evenAndOddHeaders === null ? true : $evenAndOddHeaders;
    }

    /**
     * Get the Visibility of Annotation Types
     *
     * @return \PhpOffice\PhpWord\ComplexType\TrackChangesView
     */
    public function getRevisionView()
    {
        return $this->revisionView;
    }

    /**
     * Set the Visibility of Annotation Types
     *
     * @param TrackChangesView $trackChangesView
     */
    public function setRevisionView(TrackChangesView $trackChangesView = null)
    {
        $this->revisionView = $trackChangesView;
    }

    /**
     * @return bool
     */
    public function hasTrackRevisions()
    {
        return $this->trackRevisions;
    }

    /**
     * @param bool $trackRevisions
     */
    public function setTrackRevisions($trackRevisions)
    {
        $this->trackRevisions = $trackRevisions === null ? true : $trackRevisions;
    }

    /**
     * @return bool
     */
    public function hasDoNotTrackMoves()
    {
        return $this->doNotTrackMoves;
    }

    /**
     * @param bool $doNotTrackMoves
     */
    public function setDoNotTrackMoves($doNotTrackMoves)
    {
        $this->doNotTrackMoves = $doNotTrackMoves === null ? true : $doNotTrackMoves;
    }

    /**
     * @return bool
     */
    public function hasDoNotTrackFormatting()
    {
        return $this->doNotTrackFormatting;
    }

    /**
     * @param bool $doNotTrackFormatting
     */
    public function setDoNotTrackFormatting($doNotTrackFormatting)
    {
        $this->doNotTrackFormatting = $doNotTrackFormatting === null ? true : $doNotTrackFormatting;
    }

    /**
     * @return mixed
     */
    public function getZoom()
    {
        return $this->zoom;
    }

    /**
     * @param mixed $zoom
     */
    public function setZoom($zoom)
    {
        if (is_numeric($zoom)) {
            // zoom is a percentage
            $this->zoom = $zoom;
        } else {
            Zoom::validate($zoom);
            $this->zoom = $zoom;
        }
    }

    /**
     * @return bool
     */
    public function hasMirrorMargins()
    {
        return $this->mirrorMargins;
    }

    /**
     * @param bool $mirrorMargins
     */
    public function setMirrorMargins($mirrorMargins)
    {
        $this->mirrorMargins = $mirrorMargins;
    }

    /**
     * Returns the Language
     *
     * @return Language
     */
    public function getThemeFontLang()
    {
        return $this->themeFontLang;
    }

    /**
     * sets the Language for this document
     *
     * @param Language $themeFontLang
     */
    public function setThemeFontLang($themeFontLang)
    {
        $this->themeFontLang = $themeFontLang;
    }

    /**
     * @return bool
     */
    public function hasUpdateFields()
    {
        return $this->updateFields;
    }

    /**
     * @param bool $updateFields
     */
    public function setUpdateFields($updateFields)
    {
        $this->updateFields = $updateFields === null ? false : $updateFields;
    }

    /**
     * Returns the Radix Point for Field Code Evaluation
     *
     * @return string
     */
    public function getDecimalSymbol()
    {
        return $this->decimalSymbol;
    }

    /**
     * sets the Radix Point for Field Code Evaluation
     *
     * @param string $decimalSymbol
     */
    public function setDecimalSymbol($decimalSymbol)
    {
        $this->decimalSymbol = $decimalSymbol;
    }

    /**
     * @return bool|null
     */
    public function hasAutoHyphenation()
    {
        return $this->autoHyphenation;
    }

    /**
     * @param bool $autoHyphenation
     */
    public function setAutoHyphenation($autoHyphenation)
    {
        $this->autoHyphenation = (bool) $autoHyphenation;
    }

    /**
     * @return int|null
     */
    public function getConsecutiveHyphenLimit()
    {
        return $this->consecutiveHyphenLimit;
    }

    /**
     * @param int $consecutiveHyphenLimit
     */
    public function setConsecutiveHyphenLimit($consecutiveHyphenLimit)
    {
        $this->consecutiveHyphenLimit = (int) $consecutiveHyphenLimit;
    }

    /**
     * @return float|null
     */
    public function getHyphenationZone()
    {
        return $this->hyphenationZone;
    }

    /**
     * @param float $hyphenationZone Measurement unit is twip
     */
    public function setHyphenationZone($hyphenationZone)
    {
        $this->hyphenationZone = $hyphenationZone;
    }

    /**
     * @return null|bool
     */
    public function hasDoNotHyphenateCaps()
    {
        return $this->doNotHyphenateCaps;
    }

    /**
     * @param bool $doNotHyphenateCaps
     */
    public function setDoNotHyphenateCaps($doNotHyphenateCaps)
    {
        $this->doNotHyphenateCaps = (bool) $doNotHyphenateCaps;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/PhpWord.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
*/

namespace PhpOffice\PhpWord;

use PhpOffice\PhpWord\Element\Section;
use PhpOffice\PhpWord\Exception\Exception;

/**
 * PHPWord main class
 *
 * @method Collection\Titles getTitles()
 * @method Collection\Footnotes getFootnotes()
 * @method Collection\Endnotes getEndnotes()
 * @method Collection\Charts getCharts()
 * @method Collection\Comments getComments()
 * @method int addBookmark(Element\Bookmark $bookmark)
 * @method int addTitle(Element\Title $title)
 * @method int addFootnote(Element\Footnote $footnote)
 * @method int addEndnote(Element\Endnote $endnote)
 * @method int addChart(Element\Chart $chart)
 * @method int addComment(Element\Comment $comment)
 *
 * @method Style\Paragraph addParagraphStyle(string $styleName, mixed $styles)
 * @method Style\Font addFontStyle(string $styleName, mixed $fontStyle, mixed $paragraphStyle = null)
 * @method Style\Font addLinkStyle(string $styleName, mixed $styles)
 * @method Style\Font addTitleStyle(mixed $depth, mixed $fontStyle, mixed $paragraphStyle = null)
 * @method Style\Table addTableStyle(string $styleName, mixed $styleTable, mixed $styleFirstRow = null)
 * @method Style\Numbering addNumberingStyle(string $styleName, mixed $styles)
 */
class PhpWord
{
    /**
     * Default font settings
     *
     * @deprecated 0.11.0 Use Settings constants
     *
     * @const string|int
     */
    const DEFAULT_FONT_NAME = Settings::DEFAULT_FONT_NAME;
    /**
     * @deprecated 0.11.0 Use Settings constants
     */
    const DEFAULT_FONT_SIZE = Settings::DEFAULT_FONT_SIZE;
    /**
     * @deprecated 0.11.0 Use Settings constants
     */
    const DEFAULT_FONT_COLOR = Settings::DEFAULT_FONT_COLOR;
    /**
     * @deprecated 0.11.0 Use Settings constants
     */
    const DEFAULT_FONT_CONTENT_TYPE = Settings::DEFAULT_FONT_CONTENT_TYPE;

    /**
     * Collection of sections
     *
     * @var \PhpOffice\PhpWord\Element\Section[]
     */
    private $sections = array();

    /**
     * Collections
     *
     * @var array
     */
    private $collections = array();

    /**
     * Metadata
     *
     * @var array
     * @since 0.12.0
     */
    private $metadata = array();

    /**
     * Create new instance
     *
     * Collections are created dynamically
     */
    public function __construct()
    {
        // Reset Media and styles
        Media::resetElements();
        Style::resetStyles();

        // Collection
        $collections = array('Bookmarks', 'Titles', 'Footnotes', 'Endnotes', 'Charts', 'Comments');
        foreach ($collections as $collection) {
            $class = 'PhpOffice\\PhpWord\\Collection\\' . $collection;
            $this->collections[$collection] = new $class();
        }

        // Metadata
        $metadata = array('DocInfo', 'Settings', 'Compatibility');
        foreach ($metadata as $meta) {
            $class = 'PhpOffice\\PhpWord\\Metadata\\' . $meta;
            $this->metadata[$meta] = new $class();
        }
    }

    /**
     * Dynamic function call to reduce static dependency
     *
     * @since 0.12.0
     *
     * @param mixed $function
     * @param mixed $args
     *
     * @throws \BadMethodCallException
     *
     * @return mixed
     */
    public function __call($function, $args)
    {
        $function = strtolower($function);

        $getCollection = array();
        $addCollection = array();
        $addStyle = array();

        $collections = array('Bookmark', 'Title', 'Footnote', 'Endnote', 'Chart', 'Comment');
        foreach ($collections as $collection) {
            $getCollection[] = strtolower("get{$collection}s");
            $addCollection[] = strtolower("add{$collection}");
        }

        $styles = array('Paragraph', 'Font', 'Table', 'Numbering', 'Link', 'Title');
        foreach ($styles as $style) {
            $addStyle[] = strtolower("add{$style}Style");
        }

        // Run get collection method
        if (in_array($function, $getCollection)) {
            $key = ucfirst(str_replace('get', '', $function));

            return $this->collections[$key];
        }

        // Run add collection item method
        if (in_array($function, $addCollection)) {
            $key = ucfirst(str_replace('add', '', $function) . 's');

            /** @var \PhpOffice\PhpWord\Collection\AbstractCollection $collectionObject */
            $collectionObject = $this->collections[$key];

            return $collectionObject->addItem(isset($args[0]) ? $args[0] : null);
        }

        // Run add style method
        if (in_array($function, $addStyle)) {
            return forward_static_call_array(array('PhpOffice\\PhpWord\\Style', $function), $args);
        }

        // Exception
        throw new \BadMethodCallException("Method $function is not defined.");
    }

    /**
     * Get document properties object
     *
     * @return \PhpOffice\PhpWord\Metadata\DocInfo
     */
    public function getDocInfo()
    {
        return $this->metadata['DocInfo'];
    }

    /**
     * Get protection
     *
     * @return \PhpOffice\PhpWord\Metadata\Protection
     * @since 0.12.0
     * @deprecated Get the Document protection from PhpWord->getSettings()->getDocumentProtection();
     * @codeCoverageIgnore
     */
    public function getProtection()
    {
        return $this->getSettings()->getDocumentProtection();
    }

    /**
     * Get compatibility
     *
     * @return \PhpOffice\PhpWord\Metadata\Compatibility
     * @since 0.12.0
     */
    public function getCompatibility()
    {
        return $this->metadata['Compatibility'];
    }

    /**
     * Get compatibility
     *
     * @return \PhpOffice\PhpWord\Metadata\Settings
     * @since 0.14.0
     */
    public function getSettings()
    {
        return $this->metadata['Settings'];
    }

    /**
     * Get all sections
     *
     * @return \PhpOffice\PhpWord\Element\Section[]
     */
    public function getSections()
    {
        return $this->sections;
    }

    /**
     * Returns the section at the requested position
     *
     * @param int $index
     * @return \PhpOffice\PhpWord\Element\Section|null
     */
    public function getSection($index)
    {
        if (array_key_exists($index, $this->sections)) {
            return $this->sections[$index];
        }

        return null;
    }

    /**
     * Create new section
     *
     * @param array $style
     * @return \PhpOffice\PhpWord\Element\Section
     */
    public function addSection($style = null)
    {
        $section = new Section(count($this->sections) + 1, $style);
        $section->setPhpWord($this);
        $this->sections[] = $section;

        return $section;
    }

    /**
     * Sorts the sections using the callable passed
     *
     * @see http://php.net/manual/en/function.usort.php for usage
     * @param callable $sorter
     */
    public function sortSections($sorter)
    {
        usort($this->sections, $sorter);
    }

    /**
     * Get default font name
     *
     * @return string
     */
    public function getDefaultFontName()
    {
        return Settings::getDefaultFontName();
    }

    /**
     * Set default font name.
     *
     * @param string $fontName
     */
    public function setDefaultFontName($fontName)
    {
        Settings::setDefaultFontName($fontName);
    }

    /**
     * Get default font size
     *
     * @return int
     */
    public function getDefaultFontSize()
    {
        return Settings::getDefaultFontSize();
    }

    /**
     * Set default font size.
     *
     * @param int $fontSize
     */
    public function setDefaultFontSize($fontSize)
    {
        Settings::setDefaultFontSize($fontSize);
    }

    /**
     * Set default paragraph style definition to styles.xml
     *
     * @param array $styles Paragraph style definition
     * @return \PhpOffice\PhpWord\Style\Paragraph
     */
    public function setDefaultParagraphStyle($styles)
    {
        return Style::setDefaultParagraphStyle($styles);
    }

    /**
     * Load template by filename
     *
     * @deprecated 0.12.0 Use `new TemplateProcessor($documentTemplate)` instead.
     *
     * @param  string $filename Fully qualified filename
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return TemplateProcessor
     *
     * @codeCoverageIgnore
     */
    public function loadTemplate($filename)
    {
        if (file_exists($filename)) {
            return new TemplateProcessor($filename);
        }
        throw new Exception("Template file {$filename} not found.");
    }

    /**
     * Save to file or download
     *
     * All exceptions should already been handled by the writers
     *
     * @param string $filename
     * @param string $format
     * @param bool $download
     * @return bool
     */
    public function save($filename, $format = 'Word2007', $download = false)
    {
        $mime = array(
            'Word2007'  => 'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
            'ODText'    => 'application/vnd.oasis.opendocument.text',
            'RTF'       => 'application/rtf',
            'HTML'      => 'text/html',
            'PDF'       => 'application/pdf',
        );

        $writer = IOFactory::createWriter($this, $format);

        if ($download === true) {
            header('Content-Description: File Transfer');
            header('Content-Disposition: attachment; filename="' . $filename . '"');
            header('Content-Type: ' . $mime[$format]);
            header('Content-Transfer-Encoding: binary');
            header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
            header('Expires: 0');
            $filename = 'php://output'; // Change filename to force download
        }

        $writer->save($filename);

        return true;
    }

    /**
     * Create new section
     *
     * @deprecated 0.10.0
     *
     * @param array $settings
     *
     * @return \PhpOffice\PhpWord\Element\Section
     *
     * @codeCoverageIgnore
     */
    public function createSection($settings = null)
    {
        return $this->addSection($settings);
    }

    /**
     * Get document properties object
     *
     * @deprecated 0.12.0
     *
     * @return \PhpOffice\PhpWord\Metadata\DocInfo
     *
     * @codeCoverageIgnore
     */
    public function getDocumentProperties()
    {
        return $this->getDocInfo();
    }

    /**
     * Set document properties object
     *
     * @deprecated 0.12.0
     *
     * @param \PhpOffice\PhpWord\Metadata\DocInfo $documentProperties
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setDocumentProperties($documentProperties)
    {
        $this->metadata['Document'] = $documentProperties;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/AbstractReader.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\Exception\Exception;

/**
 * Reader abstract class
 *
 * @since 0.8.0
 *
 * @codeCoverageIgnore Abstract class
 */
abstract class AbstractReader implements ReaderInterface
{
    /**
     * Read data only?
     *
     * @var bool
     */
    protected $readDataOnly = true;

    /**
     * File pointer
     *
     * @var bool|resource
     */
    protected $fileHandle;

    /**
     * Read data only?
     *
     * @return bool
     */
    public function isReadDataOnly()
    {
        // return $this->readDataOnly;
        return true;
    }

    /**
     * Set read data only
     *
     * @param bool $value
     * @return self
     */
    public function setReadDataOnly($value = true)
    {
        $this->readDataOnly = $value;

        return $this;
    }

    /**
     * Open file for reading
     *
     * @param string $filename
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return resource
     */
    protected function openFile($filename)
    {
        // Check if file exists
        if (!file_exists($filename) || !is_readable($filename)) {
            throw new Exception("Could not open $filename for reading! File does not exist.");
        }

        // Open file
        $this->fileHandle = fopen($filename, 'r');
        if ($this->fileHandle === false) {
            throw new Exception("Could not open file $filename for reading.");
        }
    }

    /**
     * Can the current ReaderInterface read the file?
     *
     * @param string $filename
     * @return bool
     */
    public function canRead($filename)
    {
        // Check if file exists
        try {
            $this->openFile($filename);
        } catch (Exception $e) {
            return false;
        }
        if (is_resource($this->fileHandle)) {
            fclose($this->fileHandle);
        }

        return true;
    }

    /**
     * Read data only?
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getReadDataOnly()
    {
        return $this->isReadDataOnly();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/HTML.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\Html as HTMLParser;

/**
 * HTML Reader class
 *
 * @since 0.11.0
 */
class HTML extends AbstractReader implements ReaderInterface
{
    /**
     * Loads PhpWord from file
     *
     * @param string $docFile
     *
     * @throws \Exception
     *
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function load($docFile)
    {
        $phpWord = new PhpWord();

        if ($this->canRead($docFile)) {
            $section = $phpWord->addSection();
            HTMLParser::addHtml($section, file_get_contents($docFile), true);
        } else {
            throw new \Exception("Cannot read {$docFile}.");
        }

        return $phpWord;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/MsDoc.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\Drawing;
use PhpOffice\PhpWord\Shared\OLERead;
use PhpOffice\PhpWord\Style;

/**
 * Reader for Word97
 *
 * @since 0.10.0
 */
class MsDoc extends AbstractReader implements ReaderInterface
{
    /**
     * PhpWord object
     *
     * @var PhpWord
     */
    private $phpWord;

    /**
     * WordDocument Stream
     *
     * @var
     */
    private $dataWorkDocument;
    /**
     * 1Table Stream
     *
     * @var
     */
    private $data1Table;
    /**
     * Data Stream
     *
     * @var
     */
    private $dataData;
    /**
     * Object Pool Stream
     *
     * @var
     */
    private $dataObjectPool;
    /**
     * @var \stdClass[]
     */
    private $arrayCharacters = array();
    /**
     * @var array
     */
    private $arrayFib = array();
    /**
     * @var string[]
     */
    private $arrayFonts = array();
    /**
     * @var string[]
     */
    private $arrayParagraphs = array();
    /**
     * @var \stdClass[]
     */
    private $arraySections = array();

    const VERSION_97 = '97';
    const VERSION_2000 = '2000';
    const VERSION_2002 = '2002';
    const VERSION_2003 = '2003';
    const VERSION_2007 = '2007';

    const SPRA_VALUE = 10;
    const SPRA_VALUE_OPPOSITE = 20;

    const OFFICEARTBLIPEMF = 0xF01A;
    const OFFICEARTBLIPWMF = 0xF01B;
    const OFFICEARTBLIPPICT = 0xF01C;
    const OFFICEARTBLIPJPG = 0xF01D;
    const OFFICEARTBLIPPNG = 0xF01E;
    const OFFICEARTBLIPDIB = 0xF01F;
    const OFFICEARTBLIPTIFF = 0xF029;
    const OFFICEARTBLIPJPEG = 0xF02A;

    const MSOBLIPERROR = 0x00;
    const MSOBLIPUNKNOWN = 0x01;
    const MSOBLIPEMF = 0x02;
    const MSOBLIPWMF = 0x03;
    const MSOBLIPPICT = 0x04;
    const MSOBLIPJPEG = 0x05;
    const MSOBLIPPNG = 0x06;
    const MSOBLIPDIB = 0x07;
    const MSOBLIPTIFF = 0x11;
    const MSOBLIPCMYKJPEG = 0x12;

    /**
     * Loads PhpWord from file
     *
     * @param string $filename
     * @return PhpWord
     */
    public function load($filename)
    {
        $this->phpWord = new PhpWord();

        $this->loadOLE($filename);

        $this->readFib($this->dataWorkDocument);
        $this->readFibContent();

        return $this->phpWord;
    }

    /**
     * Load an OLE Document
     * @param string $filename
     */
    private function loadOLE($filename)
    {
        // OLE reader
        $ole = new OLERead();
        $ole->read($filename);

        // Get WorkDocument stream
        $this->dataWorkDocument = $ole->getStream($ole->wrkdocument);
        // Get 1Table stream
        $this->data1Table = $ole->getStream($ole->wrk1Table);
        // Get Data stream
        $this->dataData = $ole->getStream($ole->wrkData);
        // Get Data stream
        $this->dataObjectPool = $ole->getStream($ole->wrkObjectPool);
        // Get Summary Information data
        $this->_SummaryInformation = $ole->getStream($ole->summaryInformation);
        // Get Document Summary Information data
        $this->_DocumentSummaryInformation = $ole->getStream($ole->docSummaryInfos);
    }

    private function getNumInLcb($lcb, $iSize)
    {
        return ($lcb - 4) / (4 + $iSize);
    }

    private function getArrayCP($data, $posMem, $iNum)
    {
        $arrayCP = array();
        for ($inc = 0; $inc < $iNum; $inc++) {
            $arrayCP[$inc] = self::getInt4d($data, $posMem);
            $posMem += 4;
        }

        return $arrayCP;
    }

    /**
     * @see  http://msdn.microsoft.com/en-us/library/dd949344%28v=office.12%29.aspx
     * @see  https://igor.io/2012/09/24/binary-parsing.html
     * @param string $data
     */
    private function readFib($data)
    {
        $pos = 0;
        //----- FibBase
        // wIdent
        $pos += 2;
        // nFib
        $pos += 2;
        // unused
        $pos += 2;
        // lid : Language Identifier
        $pos += 2;
        // pnNext
        $pos += 2;

        // $mem = self::getInt2d($data, $pos);
        // $fDot = ($mem >> 15) & 1;
        // $fGlsy = ($mem >> 14) & 1;
        // $fComplex = ($mem >> 13) & 1;
        // $fHasPic = ($mem >> 12) & 1;
        // $cQuickSaves = ($mem >> 8) & bindec('1111');
        // $fEncrypted = ($mem >> 7) & 1;
        // $fWhichTblStm = ($mem >> 6) & 1;
        // $fReadOnlyRecommended = ($mem >> 5) & 1;
        // $fWriteReservation = ($mem >> 4) & 1;
        // $fExtChar = ($mem >> 3) & 1;
        // $fLoadOverride = ($mem >> 2) & 1;
        // $fFarEast = ($mem >> 1) & 1;
        // $fObfuscated = ($mem >> 0) & 1;
        $pos += 2;
        // nFibBack
        $pos += 2;
        // lKey
        $pos += 4;
        // envr
        $pos += 1;

        // $mem = self::getInt1d($data, $pos);
        // $fMac = ($mem >> 7) & 1;
        // $fEmptySpecial = ($mem >> 6) & 1;
        // $fLoadOverridePage = ($mem >> 5) & 1;
        // $reserved1 = ($mem >> 4) & 1;
        // $reserved2 = ($mem >> 3) & 1;
        // $fSpare0 = ($mem >> 0) & bindec('111');
        $pos += 1;

        // reserved3
        $pos += 2;
        // reserved4
        $pos += 2;
        // reserved5
        $pos += 4;
        // reserved6
        $pos += 4;

        //----- csw
        $pos += 2;

        //----- fibRgW
        // reserved1
        $pos += 2;
        // reserved2
        $pos += 2;
        // reserved3
        $pos += 2;
        // reserved4
        $pos += 2;
        // reserved5
        $pos += 2;
        // reserved6
        $pos += 2;
        // reserved7
        $pos += 2;
        // reserved8
        $pos += 2;
        // reserved9
        $pos += 2;
        // reserved10
        $pos += 2;
        // reserved11
        $pos += 2;
        // reserved12
        $pos += 2;
        // reserved13
        $pos += 2;
        // lidFE
        $pos += 2;

        //----- cslw
        $pos += 2;

        //----- fibRgLw
        // cbMac
        $pos += 4;
        // reserved1
        $pos += 4;
        // reserved2
        $pos += 4;
        $this->arrayFib['ccpText'] = self::getInt4d($data, $pos);
        $pos += 4;
        $this->arrayFib['ccpFtn'] = self::getInt4d($data, $pos);
        $pos += 4;
        $this->arrayFib['ccpHdd'] = self::getInt4d($data, $pos);
        $pos += 4;
        // reserved3
        $pos += 4;
        // ccpAtn
        $pos += 4;
        // ccpEdn
        $pos += 4;
        // ccpTxbx
        $pos += 4;
        // ccpHdrTxbx
        $pos += 4;
        // reserved4
        $pos += 4;
        // reserved5
        $pos += 4;
        // reserved6
        $pos += 4;
        // reserved7
        $pos += 4;
        // reserved8
        $pos += 4;
        // reserved9
        $pos += 4;
        // reserved10
        $pos += 4;
        // reserved11
        $pos += 4;
        // reserved12
        $pos += 4;
        // reserved13
        $pos += 4;
        // reserved14
        $pos += 4;

        //----- cbRgFcLcb
        $cbRgFcLcb = self::getInt2d($data, $pos);
        $pos += 2;
        //----- fibRgFcLcbBlob
        switch ($cbRgFcLcb) {
            case 0x005D:
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_97);
                break;
            case 0x006C:
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_97);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2000);
                break;
            case 0x0088:
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_97);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2000);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2002);
                break;
            case 0x00A4:
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_97);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2000);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2002);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2003);
                break;
            case 0x00B7:
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_97);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2000);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2002);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2003);
                $pos = $this->readBlockFibRgFcLcb($data, $pos, self::VERSION_2007);
                break;
        }
        //----- cswNew
        $this->arrayFib['cswNew'] = self::getInt2d($data, $pos);
        $pos += 2;

        if ($this->arrayFib['cswNew'] != 0) {
            //@todo : fibRgCswNew
        }

        return $pos;
    }

    private function readBlockFibRgFcLcb($data, $pos, $version)
    {
        if ($version == self::VERSION_97) {
            $this->arrayFib['fcStshfOrig'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbStshfOrig'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcStshf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbStshf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcffndRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcffndRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcffndTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcffndTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfandRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfandRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfandTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfandTxt '] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfSed'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfSed'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcPad'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcPad'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfPhe'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfPhe'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfGlsy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfGlsy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfGlsy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfGlsy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfHdd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfHdd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBteChpx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBteChpx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBtePapx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBtePapx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfSea'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfSea'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfFfn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfFfn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldMom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldMom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldHdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldHdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldAtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldAtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldMcr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldMcr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmk'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmk'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcCmds'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbCmds'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfMcr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfMcr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPrDrvr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPrDrvr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPrEnvPort'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPrEnvPort'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPrEnvLand'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPrEnvLand'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcWss'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbWss'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcDop'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbDop'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfAssoc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfAssoc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcClx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbClx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfPgdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfPgdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAutosaveSource'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAutosaveSource'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcGrpXstAtnOwners'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbGrpXstAtnOwners'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfAtnBkmk'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfAtnBkmk'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused3'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused3'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcSpaMom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcSpaMom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcSpaHdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcSpaHdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfAtnBkf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfAtnBkf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfAtnBkl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfAtnBkl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPms'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPms'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcFormFldSttbs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbFormFldSttbs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfendRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfendRef'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfendTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfendTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused4'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused4'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcDggInfo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbDggInfo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfRMark'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfRMark'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfCaption'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfCaption'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfAutoCaption'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfAutoCaption'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfWkb'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfWkb'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfSpl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfSpl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcftxbxTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcftxbxTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfFldTxbx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfFldTxbx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfHdrtxbxTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfHdrtxbxTxt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcffldHdrTxbx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcffldHdrTxbx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcStwUser'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbStwUser'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbTtmbd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbTtmbd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcCookieData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbCookieData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdMotherOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdMotherOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdMotherOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdMotherOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdFtnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdFtnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdFtnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdFtnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdEdnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdEdnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdEdnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdEdnOldOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfIntlFld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfIntlFld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcRouteSlip'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbRouteSlip'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbSavedBy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbSavedBy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbFnm'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbFnm'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlfLst'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlfLst'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlfLfo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlfLfo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfTxbxBkd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfTxbxBkd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfTxbxHdrBkd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfTxbxHdrBkd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcDocUndoWord9'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbDocUndoWord9'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcRgbUse'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbRgbUse'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUsp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUsp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUskf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUskf'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcupcRgbUse'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcupcRgbUse'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcupcUsp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcupcUsp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbGlsyStyle'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbGlsyStyle'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlgosl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlgosl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcocx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcocx'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBteLvc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBteLvc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['dwLowDateTime'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['dwHighDateTime'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfLvcPre10'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfLvcPre10'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfAsumy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfAsumy'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfGram'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfGram'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbListNames'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbListNames'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfUssr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfUssr'] = self::getInt4d($data, $pos);
            $pos += 4;
        }
        if ($version == self::VERSION_2000) {
            $this->arrayFib['fcPlcfTch'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfTch'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcRmdThreading'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbRmdThreading'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcMid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbMid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbRgtplc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbRgtplc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcMsoEnvelope'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbMsoEnvelope'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfLad'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfLad'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcRgDofr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbRgDofr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcosl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcosl'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfCookieOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfCookieOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdMotherOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdMotherOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdMotherOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdMotherOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdFtnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdFtnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdFtnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdFtnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdEdnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdEdnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdEdnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdEdnOld'] = self::getInt4d($data, $pos);
            $pos += 4;
        }
        if ($version == self::VERSION_2002) {
            $this->arrayFib['fcUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfPgp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfPgp'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfuim'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfuim'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlfguidUim'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlfguidUim'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAtrdExtra'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAtrdExtra'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlrsid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlrsid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfcookie'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfcookie'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklFactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcFactoidData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbFactoidData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcDocUndo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbDocUndo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklFcc'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfbkmkBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfbkmkBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfbkfBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfbkfBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfbklBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfbklBPRepairs'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPmsNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPmsNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcODSO'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbODSO'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiOldXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiOldXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiNewXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiNewXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiMixedXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiMixedXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcffactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcffactoid'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcOldXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcOldXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcNewXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcNewXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcMixedXP'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcMixedXP'] = self::getInt4d($data, $pos);
            $pos += 4;
        }
        if ($version == self::VERSION_2003) {
            $this->arrayFib['fcHplxsdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbHplxsdr'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklSdt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcCustomXForm'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbCustomXForm'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklProt'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbProtUser'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbProtUser'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiOldInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiOldInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfpmiNewInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfpmiNewInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcOld'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcOldInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcOldInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcNew'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcflvcNewInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcflvcNewInline'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAfdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAfdMother'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAfdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAfdFtn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPgdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPgdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcBkdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbBkdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAfdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAfdEdn'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcAfd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbAfd'] = self::getInt4d($data, $pos);
            $pos += 4;
        }
        if ($version == self::VERSION_2007) {
            $this->arrayFib['fcPlcfmthd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfmthd'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklMoveFrom'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklMoveTo'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused1'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused2'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused3'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused3'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcSttbfBkmkArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbSttbfBkmkArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBkfArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBkfArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcPlcfBklArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbPlcfBklArto'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcArtoData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbArtoData'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused4'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused4'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused5'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused5'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcUnused6'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbUnused6'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcOssTheme'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbOssTheme'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['fcColorSchemeMapping'] = self::getInt4d($data, $pos);
            $pos += 4;
            $this->arrayFib['lcbColorSchemeMapping'] = self::getInt4d($data, $pos);
            $pos += 4;
        }

        return $pos;
    }

    private function readFibContent()
    {
        // Informations about Font
        $this->readRecordSttbfFfn();

        // Informations about page
        $this->readRecordPlcfSed();

        // reading paragraphs
        //@see  https://github.com/notmasteryet/CompoundFile/blob/ec118f354efebdee9102e41b5b7084fce81125b0/WordFileReader/WordDocument.cs#L86
        $this->readRecordPlcfBtePapx();

        // reading character formattings
        //@see  https://github.com/notmasteryet/CompoundFile/blob/ec118f354efebdee9102e41b5b7084fce81125b0/WordFileReader/WordDocument.cs#L94
        $this->readRecordPlcfBteChpx();

        $this->generatePhpWord();
    }

    /**
     * Section and information about them
     * @see  : http://msdn.microsoft.com/en-us/library/dd924458%28v=office.12%29.aspx
     */
    private function readRecordPlcfSed()
    {
        $posMem = $this->arrayFib['fcPlcfSed'];
        // PlcfSed
        // PlcfSed : aCP
        $aCP = array();
        $aCP[0] = self::getInt4d($this->data1Table, $posMem);
        $posMem += 4;
        $aCP[1] = self::getInt4d($this->data1Table, $posMem);
        $posMem += 4;

        // PlcfSed : aSed
        //@see  : http://msdn.microsoft.com/en-us/library/dd950194%28v=office.12%29.aspx
        $numSed = $this->getNumInLcb($this->arrayFib['lcbPlcfSed'], 12);

        $aSed = array();
        for ($iInc = 0; $iInc < $numSed; ++$iInc) {
            // Sed : http://msdn.microsoft.com/en-us/library/dd950982%28v=office.12%29.aspx
            // fn
            $posMem += 2;
            // fnMpr
            $aSed[$iInc] = self::getInt4d($this->data1Table, $posMem);
            $posMem += 4;
            // fnMpr
            $posMem += 2;
            // fcMpr
            $posMem += 4;
        }

        foreach ($aSed as $offsetSed) {
            // Sepx : http://msdn.microsoft.com/en-us/library/dd921348%28v=office.12%29.aspx
            $cb = self::getInt2d($this->dataWorkDocument, $offsetSed);
            $offsetSed += 2;

            $oStylePrl = $this->readPrl($this->dataWorkDocument, $offsetSed, $cb);
            $offsetSed += $oStylePrl->length;

            $this->arraySections[] = $oStylePrl;
        }
    }

    /**
     * Specifies the fonts that are used in the document
     * @see  : http://msdn.microsoft.com/en-us/library/dd943880%28v=office.12%29.aspx
     */
    private function readRecordSttbfFfn()
    {
        $posMem = $this->arrayFib['fcSttbfFfn'];

        $cData = self::getInt2d($this->data1Table, $posMem);
        $posMem += 2;
        $cbExtra = self::getInt2d($this->data1Table, $posMem);
        $posMem += 2;

        if ($cData < 0x7FF0 && $cbExtra == 0) {
            for ($inc = 0; $inc < $cData; $inc++) {
                // len
                $posMem += 1;
                // ffid
                $posMem += 1;
                // wWeight (400 : Normal - 700 bold)
                $posMem += 2;
                // chs
                $posMem += 1;
                // ixchSzAlt
                $ixchSzAlt = self::getInt1d($this->data1Table, $posMem);
                $posMem += 1;
                // panose
                $posMem += 10;
                // fs
                $posMem += 24;
                // xszFfn
                $xszFfn = '';
                do {
                    $char = self::getInt2d($this->data1Table, $posMem);
                    $posMem += 2;
                    if ($char > 0) {
                        $xszFfn .= chr($char);
                    }
                } while ($char != 0);
                // xszAlt
                $xszAlt = '';
                if ($ixchSzAlt > 0) {
                    do {
                        $char = self::getInt2d($this->data1Table, $posMem);
                        $posMem += 2;
                        if ($char == 0) {
                            break;
                        }
                        $xszAlt .= chr($char);
                    } while ($char != 0);
                }
                $this->arrayFonts[] = array(
                    'main' => $xszFfn,
                    'alt'  => $xszAlt,
                );
            }
        }
    }

    /**
     * Paragraph and information about them
     * @see  http://msdn.microsoft.com/en-us/library/dd908569%28v=office.12%29.aspx
     */
    private function readRecordPlcfBtePapx()
    {
        $posMem = $this->arrayFib['fcPlcfBtePapx'];
        $num = $this->getNumInLcb($this->arrayFib['lcbPlcfBtePapx'], 4);
        $posMem += 4 * ($num + 1);
        $arrAPnBtePapx = $this->getArrayCP($this->data1Table, $posMem, $num);
        $posMem += 4 * $num;

        foreach ($arrAPnBtePapx as $aPnBtePapx) {
            $offsetBase = $aPnBtePapx * 512;
            $offset = $offsetBase;

            $string = '';

            $numRun = self::getInt1d($this->dataWorkDocument, $offset + 511);
            $arrayRGFC = array();
            for ($inc = 0; $inc <= $numRun; $inc++) {
                $arrayRGFC[$inc] = self::getInt4d($this->dataWorkDocument, $offset);
                $offset += 4;
            }
            $arrayRGB = array();
            for ($inc = 1; $inc <= $numRun; $inc++) {
                // @see  http://msdn.microsoft.com/en-us/library/dd925804(v=office.12).aspx
                $arrayRGB[$inc] = self::getInt1d($this->dataWorkDocument, $offset);
                $offset += 1;
                // reserved
                $offset += 12;
            }

            foreach (array_keys($arrayRGFC) as $key) {
                if (!isset($arrayRGFC[$key + 1])) {
                    break;
                }
                $strLen = $arrayRGFC[$key + 1] - $arrayRGFC[$key] - 1;
                for ($inc = 0; $inc < $strLen; $inc++) {
                    $byte = self::getInt1d($this->dataWorkDocument, $arrayRGFC[$key] + $inc);
                    if ($byte > 0) {
                        $string .= chr($byte);
                    }
                }
            }
            $this->arrayParagraphs[] = $string;

            //@todo readPrl for paragraphs
            /*// use $this->readPrl()
            foreach ($arrayRGB as $key => $rgb) {
                $offset = $offsetBase + ($rgb * 2);

                $cb = self::getInt1d($this->dataWorkDocument, $offset);
                $offset += 1;
                print_r('$cb : '.$cb.PHP_EOL);
                if ($cb == 0) {
                    $cb = self::getInt1d($this->dataWorkDocument, $offset);
                    $cb = $cb * 2;
                    $offset += 1;
                    print_r('$cb0 : '.$cb.PHP_EOL);
                } else {
                    $cb = $cb * 2 - 1;
                    print_r('$cbD : '.$cb.PHP_EOL);
                }
                $istd = self::getInt2d($this->dataWorkDocument, $offset);
                $offset += 2;
                $cb -= 2;
                print_r('$istd : '.$istd.($istd == 0 ? ' (Normal)' : '').PHP_EOL);
                if ($cb > 0) {
                    do{
                        $sprm = self::getInt2d($this->dataWorkDocument, $offset);
                        $offset += 2;
                        $cb -= 2;
                        $sprm_IsPmd = $sprm & 0x01FF;
                        $sprm_F = ($sprm/512) & 0x0001;
                        $sprm_Sgc = ($sprm/1024) & 0x0007;
                        $sprm_Spra = ($sprm/8192);

                        print_r('$sprm : 0x'.dechex($sprm).PHP_EOL);
                        print_r('$sprm.ispmd : 0x'.dechex($sprm_IsPmd).PHP_EOL);
                        print_r('$sprm.f : 0x'.dechex($sprm_F).PHP_EOL);
                        print_r('$sprm.sgc : 0x'.dechex($sprm_Sgc));
                        switch (dechex($sprm_Sgc)) {
                            case 0x01:
                                print_r(' (Paragraph property)');
                                break;
                            case 0x02:
                                print_r(' (Character property)');
                                break;
                            case 0x03:
                                print_r(' (Picture property)');
                                break;
                            case 0x04:
                                print_r(' (Section property)');
                                break;
                            case 0x05:
                                print_r(' (Table property)');
                                break;
                        }
                        print_r(PHP_EOL);
                        print_r('$sprm.spra : 0x'.dechex($sprm_Spra).PHP_EOL);
                        switch (dechex($sprm_Spra)) {
                            case 0x0:
                                $operand = self::getInt1d($this->dataWorkDocument, $offset);
                                $offset += 1;
                                $cb -= 1;
                                switch (dechex($operand)) {
                                    case 0x00:
                                        $operand = 'OFF';
                                        break;
                                    case 0x01:
                                        $operand = 'ON';
                                        break;
                                    case 0x80:
                                        $operand = 'CURRENT VALUE';
                                        print_r(''.PHP_EOL.PHP_EOL);
                                        break;
                                    case 0x81:
                                        $operand = 'OPPOSITE OF THE CURRENT VALUE';
                                        break;
                                }
                                break;
                            case 0x1:
                                $operand = self::getInt1d($this->dataWorkDocument, $offset);
                                $offset += 1;
                                $cb -= 1;
                                print_r('$operand : 0x'.dechex($operand).PHP_EOL);
                                break;
                            case 0x2:
                            case 0x4:
                            case 0x5:
                                $operand = self::getInt2d($this->dataWorkDocument, $offset);
                                $offset += 2;
                                $cb -= 2;
                                print_r('$operand : 0x'.dechex($operand).PHP_EOL);
                                break;
                            case 0x3:
                                if ($sprm_IsPmd != 0x70) {
                                    $operand = self::getInt4d($this->dataWorkDocument, $offset);
                                    $offset += 4;
                                    $cb -= 4;
                                    print_r('$operand : 0x'.dechex($operand).PHP_EOL);
                                }
                                break;
                            case 0x7:
                                $operand = self::getInt3d($this->dataWorkDocument, $offset);
                                $offset += 3;
                                $cb -= 3;
                                print_r('$operand : 0x'.dechex($operand).PHP_EOL);
                                break;
                            default:
                                print_r('YO YO YO : '.PHP_EOL);
                        }

                        //
                        switch (dechex($sprm_Sgc)) {
                            case 0x01: // Sprm is modifying a paragraph property.
                                switch ($sprm_IsPmd) {
                                    case 0x0A: // sprmPIlvl
                                        print_r('sprmPIlvl : '.$operand.PHP_EOL.PHP_EOL);
                                        break;
                                    case 0x0B: // sprmPIlfo
                                        print_r('sprmPIlfo : '.$operand.PHP_EOL.PHP_EOL);
                                        break;
                                    default:
                                        print_r('$sprm_IsPmd(1) : '.$sprm_IsPmd.PHP_EOL.PHP_EOL);
                                        break;
                                }
                                break;
                            case 0x02: // Sprm is modifying a character property.
                                switch ($sprm_IsPmd) {
                                    default:
                                        print_r('$sprm_IsPmd(2) : '.$sprm_IsPmd.PHP_EOL.PHP_EOL);
                                        break;
                                }
                                break;
                            case 0x03: // Sprm is modifying a picture property.
                                switch ($sprm_IsPmd) {
                                    default:
                                        print_r('$sprm_IsPmd(3) : '.$sprm_IsPmd.PHP_EOL.PHP_EOL);
                                        break;
                                }
                                break;
                            case 0x04: // Sprm is modifying a section property.
                                switch ($sprm_IsPmd) {
                                    default:
                                        print_r('$sprm_IsPmd(4) : '.$sprm_IsPmd.PHP_EOL.PHP_EOL);
                                        break;
                                }
                                break;
                            case 0x05: // Sprm is modifying a table property.
                                switch ($sprm_IsPmd) {
                                    default:
                                        print_r('$sprm_IsPmd(4) : '.$sprm_IsPmd.PHP_EOL.PHP_EOL);
                                        break;
                                }
                                break;
                            default:
                                print_r('$sprm_Sgc : '.dechex($sprm_Sgc).PHP_EOL.PHP_EOL);
                                break;
                        }
                    } while ($cb > 0);
                } else {
                    if ($istd > 0) {
                        // @todo : Determining Properties of a Paragraph Style
                        # @see  http://msdn.microsoft.com/en-us/library/dd948631%28v=office.12%29.aspx
                    }
                }
            }*/
        }
    }

    /**
     * Character formatting properties to text in a document
     * @see  http://msdn.microsoft.com/en-us/library/dd907108%28v=office.12%29.aspx
     */
    private function readRecordPlcfBteChpx()
    {
        $posMem = $this->arrayFib['fcPlcfBteChpx'];
        $num = $this->getNumInLcb($this->arrayFib['lcbPlcfBteChpx'], 4);
        $aPnBteChpx = array();
        for ($inc = 0; $inc <= $num; $inc++) {
            $aPnBteChpx[$inc] = self::getInt4d($this->data1Table, $posMem);
            $posMem += 4;
        }
        $pnFkpChpx = self::getInt4d($this->data1Table, $posMem);
        $posMem += 4;

        $offsetBase = $pnFkpChpx * 512;
        $offset = $offsetBase;

        // ChpxFkp
        // @see  : http://msdn.microsoft.com/en-us/library/dd910989%28v=office.12%29.aspx
        $numRGFC = self::getInt1d($this->dataWorkDocument, $offset + 511);
        $arrayRGFC = array();
        for ($inc = 0; $inc <= $numRGFC; $inc++) {
            $arrayRGFC[$inc] = self::getInt4d($this->dataWorkDocument, $offset);
            $offset += 4;
        }

        $arrayRGB = array();
        for ($inc = 1; $inc <= $numRGFC; $inc++) {
            $arrayRGB[$inc] = self::getInt1d($this->dataWorkDocument, $offset);
            $offset += 1;
        }

        $start = 0;
        foreach ($arrayRGB as $keyRGB => $rgb) {
            $oStyle = new \stdClass();
            $oStyle->pos_start = $start;
            $oStyle->pos_len = (int) ceil((($arrayRGFC[$keyRGB] - 1) - $arrayRGFC[$keyRGB - 1]) / 2);
            $start += $oStyle->pos_len;

            if ($rgb > 0) {
                // Chp Structure
                // @see  : http://msdn.microsoft.com/en-us/library/dd772849%28v=office.12%29.aspx
                $posRGB = $offsetBase + $rgb * 2;

                $cb = self::getInt1d($this->dataWorkDocument, $posRGB);
                $posRGB += 1;

                $oStyle->style = $this->readPrl($this->dataWorkDocument, $posRGB, $cb);
                $posRGB += $oStyle->style->length;
            }
            $this->arrayCharacters[] = $oStyle;
        }
    }

    /**
     * @param $sprm
     * @return \stdClass
     */
    private function readSprm($sprm)
    {
        $oSprm = new \stdClass();
        $oSprm->isPmd = $sprm & 0x01FF;
        $oSprm->f = ($sprm / 512) & 0x0001;
        $oSprm->sgc = ($sprm / 1024) & 0x0007;
        $oSprm->spra = ($sprm / 8192);

        return $oSprm;
    }

    /**
     * @param string $data
     * @param int $pos
     * @param \stdClass $oSprm
     * @return array
     */
    private function readSprmSpra($data, $pos, $oSprm)
    {
        $length = 0;
        $operand = null;

        switch (dechex($oSprm->spra)) {
            case 0x0:
                $operand = self::getInt1d($data, $pos);
                $length = 1;
                switch (dechex($operand)) {
                    case 0x00:
                        $operand = false;
                        break;
                    case 0x01:
                        $operand = true;
                        break;
                    case 0x80:
                        $operand = self::SPRA_VALUE;
                        break;
                    case 0x81:
                        $operand = self::SPRA_VALUE_OPPOSITE;
                        break;
                }
                break;
            case 0x1:
                $operand = self::getInt1d($data, $pos);
                $length = 1;
                break;
            case 0x2:
            case 0x4:
            case 0x5:
                $operand = self::getInt2d($data, $pos);
                $length = 2;
                break;
            case 0x3:
                if ($oSprm->isPmd != 0x70) {
                    $operand = self::getInt4d($data, $pos);
                    $length = 4;
                }
                break;
            case 0x7:
                $operand = self::getInt3d($data, $pos);
                $length = 3;
                break;
            default:
                // print_r('YO YO YO : '.PHP_EOL);
        }

        return array(
            'length'  => $length,
            'operand' => $operand,
        );
    }

    /**
     * @param $data int
     * @param $pos int
     * @param $cbNum int
     * @return \stdClass
     * @see  http://msdn.microsoft.com/en-us/library/dd772849%28v=office.12%29.aspx
     */
    private function readPrl($data, $pos, $cbNum)
    {
        $posStart = $pos;
        $oStylePrl = new \stdClass();

        // Variables
        $sprmCPicLocation = null;
        $sprmCFData = null;
        //$sprmCFSpec = null;

        do {
            // Variables
            $operand = null;

            $sprm = self::getInt2d($data, $pos);
            $oSprm = $this->readSprm($sprm);
            $pos += 2;
            $cbNum -= 2;

            $arrayReturn = $this->readSprmSpra($data, $pos, $oSprm);
            $pos += $arrayReturn['length'];
            $cbNum -= $arrayReturn['length'];
            $operand = $arrayReturn['operand'];

            switch (dechex($oSprm->sgc)) {
                // Paragraph property
                case 0x01:
                    break;
                // Character property
                case 0x02:
                    if (!isset($oStylePrl->styleFont)) {
                        $oStylePrl->styleFont = array();
                    }
                    switch ($oSprm->isPmd) {
                        // sprmCFRMarkIns
                        case 0x01:
                            break;
                        // sprmCFFldVanish
                        case 0x02:
                            break;
                        // sprmCPicLocation
                        case 0x03:
                            $sprmCPicLocation = $operand;
                            break;
                        // sprmCFData
                        case 0x06:
                            $sprmCFData = dechex($operand) != 0x00;
                            break;
                        // sprmCFItalic
                        case 0x36:
                            // By default, text is not italicized.
                            switch ($operand) {
                                case false:
                                case true:
                                    $oStylePrl->styleFont['italic'] = $operand;
                                    break;
                                case self::SPRA_VALUE:
                                    $oStylePrl->styleFont['italic'] = false;
                                    break;
                                case self::SPRA_VALUE_OPPOSITE:
                                    $oStylePrl->styleFont['italic'] = true;
                                    break;
                            }
                            break;
                        // sprmCIstd
                        case 0x30:
                            //print_r('sprmCIstd : '.dechex($operand).PHP_EOL.PHP_EOL);
                            break;
                        // sprmCFBold
                        case 0x35:
                            // By default, text is not bold.
                            switch ($operand) {
                                case false:
                                case true:
                                    $oStylePrl->styleFont['bold'] = $operand;
                                    break;
                                case self::SPRA_VALUE:
                                    $oStylePrl->styleFont['bold'] = false;
                                    break;
                                case self::SPRA_VALUE_OPPOSITE:
                                    $oStylePrl->styleFont['bold'] = true;
                                    break;
                            }
                            break;
                        // sprmCFStrike
                        case 0x37:
                            // By default, text is not struck through.
                            switch ($operand) {
                                case false:
                                case true:
                                    $oStylePrl->styleFont['strikethrough'] = $operand;
                                    break;
                                case self::SPRA_VALUE:
                                    $oStylePrl->styleFont['strikethrough'] = false;
                                    break;
                                case self::SPRA_VALUE_OPPOSITE:
                                    $oStylePrl->styleFont['strikethrough'] = true;
                                    break;
                            }
                            break;
                        // sprmCKul
                        case 0x3E:
                            switch (dechex($operand)) {
                                case 0x00:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_NONE;
                                    break;
                                case 0x01:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_SINGLE;
                                    break;
                                case 0x02:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_WORDS;
                                    break;
                                case 0x03:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOUBLE;
                                    break;
                                case 0x04:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTTED;
                                    break;
                                case 0x06:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_HEAVY;
                                    break;
                                case 0x07:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DASH;
                                    break;
                                case 0x09:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTDASH;
                                    break;
                                case 0x0A:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTDOTDASH;
                                    break;
                                case 0x0B:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_WAVY;
                                    break;
                                case 0x14:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTTEDHEAVY;
                                    break;
                                case 0x17:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DASHHEAVY;
                                    break;
                                case 0x19:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTDASHHEAVY;
                                    break;
                                case 0x1A:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DOTDOTDASHHEAVY;
                                    break;
                                case 0x1B:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_WAVYHEAVY;
                                    break;
                                case 0x27:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DASHLONG;
                                    break;
                                case 0x2B:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_WAVYDOUBLE;
                                    break;
                                case 0x37:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_DASHLONGHEAVY;
                                    break;
                                default:
                                    $oStylePrl->styleFont['underline'] = Style\Font::UNDERLINE_NONE;
                                    break;
                            }
                            break;
                        // sprmCIco
                        //@see  http://msdn.microsoft.com/en-us/library/dd773060%28v=office.12%29.aspx
                        case 0x42:
                            switch (dechex($operand)) {
                                case 0x00:
                                case 0x01:
                                    $oStylePrl->styleFont['color'] = '000000';
                                    break;
                                case 0x02:
                                    $oStylePrl->styleFont['color'] = '0000FF';
                                    break;
                                case 0x03:
                                    $oStylePrl->styleFont['color'] = '00FFFF';
                                    break;
                                case 0x04:
                                    $oStylePrl->styleFont['color'] = '00FF00';
                                    break;
                                case 0x05:
                                    $oStylePrl->styleFont['color'] = 'FF00FF';
                                    break;
                                case 0x06:
                                    $oStylePrl->styleFont['color'] = 'FF0000';
                                    break;
                                case 0x07:
                                    $oStylePrl->styleFont['color'] = 'FFFF00';
                                    break;
                                case 0x08:
                                    $oStylePrl->styleFont['color'] = 'FFFFFF';
                                    break;
                                case 0x09:
                                    $oStylePrl->styleFont['color'] = '000080';
                                    break;
                                case 0x0A:
                                    $oStylePrl->styleFont['color'] = '008080';
                                    break;
                                case 0x0B:
                                    $oStylePrl->styleFont['color'] = '008000';
                                    break;
                                case 0x0C:
                                    $oStylePrl->styleFont['color'] = '800080';
                                    break;
                                case 0x0D:
                                    $oStylePrl->styleFont['color'] = '800080';
                                    break;
                                case 0x0E:
                                    $oStylePrl->styleFont['color'] = '808000';
                                    break;
                                case 0x0F:
                                    $oStylePrl->styleFont['color'] = '808080';
                                    break;
                                case 0x10:
                                    $oStylePrl->styleFont['color'] = 'C0C0C0';
                            }
                            break;
                        // sprmCHps
                        case 0x43:
                            $oStylePrl->styleFont['size'] = dechex($operand / 2);
                            break;
                        // sprmCIss
                        case 0x48:
                            if (!isset($oStylePrl->styleFont['superScript'])) {
                                $oStylePrl->styleFont['superScript'] = false;
                            }
                            if (!isset($oStylePrl->styleFont['subScript'])) {
                                $oStylePrl->styleFont['subScript'] = false;
                            }
                            switch (dechex($operand)) {
                                case 0x00:
                                    // Normal text
                                    break;
                                case 0x01:
                                    $oStylePrl->styleFont['superScript'] = true;
                                    break;
                                case 0x02:
                                    $oStylePrl->styleFont['subScript'] = true;
                                    break;
                            }
                            break;
                        // sprmCRgFtc0
                        case 0x4F:
                            $oStylePrl->styleFont['name'] = '';
                            if (isset($this->arrayFonts[$operand])) {
                                $oStylePrl->styleFont['name'] = $this->arrayFonts[$operand]['main'];
                            }
                            break;
                        // sprmCRgFtc1
                        case 0x50:
                            // if the language for the text is an East Asian language
                            break;
                        // sprmCRgFtc2
                        case 0x51:
                            // if the character falls outside the Unicode character range
                            break;
                        // sprmCFSpec
                        case 0x55:
                            //$sprmCFSpec = $operand;
                            break;
                        // sprmCFtcBi
                        case 0x5E:
                            break;
                        // sprmCFItalicBi
                        case 0x5D:
                            break;
                        // sprmCHpsBi
                        case 0x61:
                            break;
                        // sprmCShd80
                        //@see  http://msdn.microsoft.com/en-us/library/dd923447%28v=office.12%29.aspx
                        case 0x66:
                            // $operand = self::getInt2d($data, $pos);
                            $pos += 2;
                            $cbNum -= 2;
                            // $ipat = ($operand >> 0) && bindec('111111');
                            // $icoBack = ($operand >> 6) && bindec('11111');
                            // $icoFore = ($operand >> 11) && bindec('11111');
                            break;
                        // sprmCCv
                        //@see  : http://msdn.microsoft.com/en-us/library/dd952824%28v=office.12%29.aspx
                        case 0x70:
                            $red = str_pad(dechex(self::getInt1d($this->dataWorkDocument, $pos)), 2, '0', STR_PAD_LEFT);
                            $pos += 1;
                            $green = str_pad(dechex(self::getInt1d($this->dataWorkDocument, $pos)), 2, '0', STR_PAD_LEFT);
                            $pos += 1;
                            $blue = str_pad(dechex(self::getInt1d($this->dataWorkDocument, $pos)), 2, '0', STR_PAD_LEFT);
                            $pos += 1;
                            $pos += 1;
                            $oStylePrl->styleFont['color'] = $red . $green . $blue;
                            $cbNum -= 4;
                            break;
                        default:
                            // print_r('@todo Character : 0x'.dechex($oSprm->isPmd));
                            // print_r(PHP_EOL);
                    }
                    break;
                // Picture property
                case 0x03:
                    break;
                // Section property
                case 0x04:
                    if (!isset($oStylePrl->styleSection)) {
                        $oStylePrl->styleSection = array();
                    }
                    switch ($oSprm->isPmd) {
                        // sprmSNfcPgn
                        case 0x0E:
                            // numbering format used for page numbers
                            break;
                        // sprmSXaPage
                        case 0x1F:
                            $oStylePrl->styleSection['pageSizeW'] = $operand;
                            break;
                        // sprmSYaPage
                        case 0x20:
                            $oStylePrl->styleSection['pageSizeH'] = $operand;
                            break;
                        // sprmSDxaLeft
                        case 0x21:
                            $oStylePrl->styleSection['marginLeft'] = $operand;
                            break;
                        // sprmSDxaRight
                        case 0x22:
                            $oStylePrl->styleSection['marginRight'] = $operand;
                            break;
                        // sprmSDyaTop
                        case 0x23:
                            $oStylePrl->styleSection['marginTop'] = $operand;
                            break;
                        // sprmSDyaBottom
                        case 0x24:
                            $oStylePrl->styleSection['marginBottom'] = $operand;
                            break;
                        // sprmSFBiDi
                        case 0x28:
                            // RTL layout
                            break;
                        // sprmSDxtCharSpace
                        case 0x30:
                            // characpter pitch
                            break;
                        // sprmSDyaLinePitch
                        case 0x31:
                            // line height
                            break;
                        // sprmSClm
                        case 0x32:
                            // document grid mode
                            break;
                        // sprmSTextFlow
                        case 0x33:
                            // text flow
                            break;
                        default:
                            // print_r('@todo Section : 0x'.dechex($oSprm->isPmd));
                            // print_r(PHP_EOL);
                    }
                    break;
                // Table property
                case 0x05:
                    break;
            }
        } while ($cbNum > 0);

        if (!is_null($sprmCPicLocation)) {
            if (!is_null($sprmCFData) && $sprmCFData == 0x01) {
                // NilPICFAndBinData
                //@todo Read Hyperlink structure
                /*$lcb = self::getInt4d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 4;
                $cbHeader = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // ignored
                $sprmCPicLocation += 62;
                // depending of the element
                // Hyperlink => HFD
                // HFD > bits
                $sprmCPicLocation += 1;
                // HFD > clsid
                $sprmCPicLocation += 16;
                // HFD > hyperlink
                //@see  : http://msdn.microsoft.com/en-us/library/dd909835%28v=office.12%29.aspx
                $streamVersion = self::getInt4d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 4;
                $data = self::getInt4d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 4;
                $hlstmfAbsFromGetdataRel = ($data >> 9) & bindec('1');
                $hlstmfMonikerSavedAsStr = ($data >> 8) & bindec('1');
                $hlstmfHasFrameName = ($data >> 7) & bindec('1');
                $hlstmfHasCreationTime = ($data >> 6) & bindec('1');
                $hlstmfHasGUID = ($data >> 5) & bindec('1');
                $hlstmfHasDisplayName = ($data >> 4) & bindec('1');
                $hlstmfHasLocationStr = ($data >> 3) & bindec('1');
                $hlstmfSiteGaveDisplayName = ($data >> 2) & bindec('1');
                $hlstmfIsAbsolute = ($data >> 1) & bindec('1');
                $hlstmfHasMoniker = ($data >> 0) & bindec('1');
                for ($inc = 0; $inc <= 32; $inc++) {
                    echo ($data >> $inc) & bindec('1');
                }

                print_r('$hlstmfHasMoniker > '.$hlstmfHasMoniker.PHP_EOL);
                print_r('$hlstmfIsAbsolute > '.$hlstmfIsAbsolute.PHP_EOL);
                print_r('$hlstmfSiteGaveDisplayName > '.$hlstmfSiteGaveDisplayName.PHP_EOL);
                print_r('$hlstmfHasLocationStr > '.$hlstmfHasLocationStr.PHP_EOL);
                print_r('$hlstmfHasDisplayName > '.$hlstmfHasDisplayName.PHP_EOL);
                print_r('$hlstmfHasGUID > '.$hlstmfHasGUID.PHP_EOL);
                print_r('$hlstmfHasCreationTime > '.$hlstmfHasCreationTime.PHP_EOL);
                print_r('$hlstmfHasFrameName > '.$hlstmfHasFrameName.PHP_EOL);
                print_r('$hlstmfMonikerSavedAsStr > '.$hlstmfMonikerSavedAsStr.PHP_EOL);
                print_r('$hlstmfAbsFromGetdataRel > '.$hlstmfAbsFromGetdataRel.PHP_EOL);
                if ($streamVersion == 2) {
                    $AAA = self::getInt4d($this->dataData, $sprmCPicLocation);
                    echo 'AAAA : '.$AAA.PHP_EOL;
                    if ($hlstmfHasDisplayName == 1) {
                        echo 'displayName'.PHP_EOL;
                    }
                    if ($hlstmfHasFrameName == 1) {
                        echo 'targetFrameName'.PHP_EOL;
                    }
                    if ($hlstmfHasMoniker == 1 || $hlstmfMonikerSavedAsStr == 1) {
                        $sprmCPicLocation += 16;
                        $length = self::getInt4d($this->dataData, $sprmCPicLocation);
                        $sprmCPicLocation += 4;
                        for ($inc = 0; $inc < ($length / 2); $inc++) {
                            $chr = self::getInt2d($this->dataData, $sprmCPicLocation);
                            $sprmCPicLocation += 2;
                            print_r(chr($chr));
                        }
                        echo PHP_EOL;
                        echo 'moniker : '.$length.PHP_EOL;
                    }
                    if ($hlstmfHasMoniker == 1 || $hlstmfMonikerSavedAsStr == 1) {
                        echo 'oleMoniker'.PHP_EOL;
                    }
                    if ($hlstmfHasLocationStr == 1) {
                        echo 'location'.PHP_EOL;
                    }
                    if ($hlstmfHasGUID == 1) {
                        echo 'guid'.PHP_EOL;
                        $sprmCPicLocation += 16;
                    }
                    if ($hlstmfHasCreationTime == 1) {
                        echo 'fileTime'.PHP_EOL;
                        $sprmCPicLocation += 4;
                    }
                    echo 'HYPERLINK'.PHP_EOL;
                }*/
            } else {
                // Pictures
                //@see  : http://msdn.microsoft.com/en-us/library/dd925458%28v=office.12%29.aspx
                //@see  : http://msdn.microsoft.com/en-us/library/dd926136%28v=office.12%29.aspx
                // PICF : lcb
                $sprmCPicLocation += 4;
                // PICF : cbHeader
                $sprmCPicLocation += 2;
                // PICF : mfpf : mm
                $mfpfMm = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : mfpf : xExt
                $sprmCPicLocation += 2;
                // PICF : mfpf : yExt
                $sprmCPicLocation += 2;
                // PICF : mfpf : swHMF
                $sprmCPicLocation += 2;
                // PICF : innerHeader : grf
                $sprmCPicLocation += 4;
                // PICF : innerHeader : padding1
                $sprmCPicLocation += 4;
                // PICF : innerHeader : mmPM
                $sprmCPicLocation += 2;
                // PICF : innerHeader : padding2
                $sprmCPicLocation += 4;
                // PICF : picmid : dxaGoal
                $picmidDxaGoal = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : dyaGoal
                $picmidDyaGoal = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : mx
                $picmidMx = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : my
                $picmidMy = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : dxaReserved1
                $picmidDxaCropLeft = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : dyaReserved1
                $picmidDxaCropTop = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : dxaReserved2
                $picmidDxaCropRight = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : dyaReserved2
                $picmidDxaCropBottom = self::getInt2d($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 2;
                // PICF : picmid : fReserved
                $sprmCPicLocation += 1;
                // PICF : picmid : bpp
                $sprmCPicLocation += 1;
                // PICF : picmid : brcTop80
                $sprmCPicLocation += 4;
                // PICF : picmid : brcLeft80
                $sprmCPicLocation += 4;
                // PICF : picmid : brcBottom80
                $sprmCPicLocation += 4;
                // PICF : picmid : brcRight80
                $sprmCPicLocation += 4;
                // PICF : picmid : dxaReserved3
                $sprmCPicLocation += 2;
                // PICF : picmid : dyaReserved3
                $sprmCPicLocation += 2;
                // PICF : cProps
                $sprmCPicLocation += 2;

                if ($mfpfMm == 0x0066) {
                    // cchPicName
                    $cchPicName = self::getInt1d($this->dataData, $sprmCPicLocation);
                    $sprmCPicLocation += 1;

                    // stPicName
                    //$stPicName = '';
                    for ($inc = 0; $inc <= $cchPicName; $inc++) {
                        //$chr = self::getInt1d($this->dataData, $sprmCPicLocation);
                        $sprmCPicLocation += 1;
                        //$stPicName .= chr($chr);
                    }
                }

                // picture (OfficeArtInlineSpContainer)
                // picture : shape
                $shapeRH = $this->loadRecordHeader($this->dataData, $sprmCPicLocation);
                $sprmCPicLocation += 8;
                if ($shapeRH['recVer'] == 0xF && $shapeRH['recInstance'] == 0x000 && $shapeRH['recType'] == 0xF004) {
                    $sprmCPicLocation += $shapeRH['recLen'];
                }
                // picture : rgfb
                //@see  : http://msdn.microsoft.com/en-us/library/dd950560%28v=office.12%29.aspx
                $fileBlockRH = $this->loadRecordHeader($this->dataData, $sprmCPicLocation);
                while ($fileBlockRH['recType'] == 0xF007 || ($fileBlockRH['recType'] >= 0xF018 && $fileBlockRH['recType'] <= 0xF117)) {
                    $sprmCPicLocation += 8;
                    switch ($fileBlockRH['recType']) {
                        // OfficeArtFBSE
                        //@see  : http://msdn.microsoft.com/en-us/library/dd944923%28v=office.12%29.aspx
                        case 0xF007:
                            // btWin32
                            $sprmCPicLocation += 1;
                            // btMacOS
                            $sprmCPicLocation += 1;
                            // rgbUid
                            $sprmCPicLocation += 16;
                            // tag
                            $sprmCPicLocation += 2;
                            // size
                            $sprmCPicLocation += 4;
                            // cRef
                            $sprmCPicLocation += 4;
                            // foDelay
                            $sprmCPicLocation += 4;
                            // unused1
                            $sprmCPicLocation += 1;
                            // cbName
                            $cbName = self::getInt1d($this->dataData, $sprmCPicLocation);
                            $sprmCPicLocation += 1;
                            // unused2
                            $sprmCPicLocation += 1;
                            // unused3
                            $sprmCPicLocation += 1;
                            // nameData
                            if ($cbName > 0) {
                                //$nameData = '';
                                for ($inc = 0; $inc <= ($cbName / 2); $inc++) {
                                    //$chr = self::getInt2d($this->dataData, $sprmCPicLocation);
                                    $sprmCPicLocation += 2;
                                    //$nameData .= chr($chr);
                                }
                            }
                            // embeddedBlip
                            //@see  : http://msdn.microsoft.com/en-us/library/dd910081%28v=office.12%29.aspx
                            $embeddedBlipRH = $this->loadRecordHeader($this->dataData, $sprmCPicLocation);
                            switch ($embeddedBlipRH['recType']) {
                                case self::OFFICEARTBLIPJPG:
                                case self::OFFICEARTBLIPJPEG:
                                    if (!isset($oStylePrl->image)) {
                                        $oStylePrl->image = array();
                                    }
                                    $sprmCPicLocation += 8;
                                    // embeddedBlip : rgbUid1
                                    $sprmCPicLocation += 16;
                                    if ($embeddedBlipRH['recInstance'] == 0x6E1) {
                                        // rgbUid2
                                        $sprmCPicLocation += 16;
                                    }
                                    // embeddedBlip : tag
                                    $sprmCPicLocation += 1;
                                    // embeddedBlip : BLIPFileData
                                    $oStylePrl->image['data'] = substr($this->dataData, $sprmCPicLocation, $embeddedBlipRH['recLen']);
                                    $oStylePrl->image['format'] = 'jpg';
                                    // Image Size
                                    $iCropWidth = $picmidDxaGoal - ($picmidDxaCropLeft + $picmidDxaCropRight);
                                    $iCropHeight = $picmidDyaGoal - ($picmidDxaCropTop + $picmidDxaCropBottom);
                                    if (!$iCropWidth) {
                                        $iCropWidth = 1;
                                    }
                                    if (!$iCropHeight) {
                                        $iCropHeight = 1;
                                    }
                                    $oStylePrl->image['width'] = Drawing::twipsToPixels($iCropWidth * $picmidMx / 1000);
                                    $oStylePrl->image['height'] = Drawing::twipsToPixels($iCropHeight * $picmidMy / 1000);

                                    $sprmCPicLocation += $embeddedBlipRH['recLen'];
                                    break;
                                case self::OFFICEARTBLIPPNG:
                                    break;
                                default:
                                    // print_r(dechex($embeddedBlipRH['recType']));
                            }
                            break;
                    }
                    $fileBlockRH = $this->loadRecordHeader($this->dataData, $sprmCPicLocation);
                }
            }
        }

        $oStylePrl->length = $pos - $posStart;

        return $oStylePrl;
    }

    /**
     * Read a record header
     * @param string $stream
     * @param int $pos
     * @return array
     */
    private function loadRecordHeader($stream, $pos)
    {
        $rec = self::getInt2d($stream, $pos);
        $recType = self::getInt2d($stream, $pos + 2);
        $recLen = self::getInt4d($stream, $pos + 4);

        return array(
            'recVer'      => ($rec >> 0) & bindec('1111'),
            'recInstance' => ($rec >> 4) & bindec('111111111111'),
            'recType'     => $recType,
            'recLen'      => $recLen,
        );
    }

    private function generatePhpWord()
    {
        foreach ($this->arraySections as $itmSection) {
            $oSection = $this->phpWord->addSection();
            $oSection->setStyle($itmSection->styleSection);

            $sHYPERLINK = '';
            foreach ($this->arrayParagraphs as $itmParagraph) {
                $textPara = $itmParagraph;
                foreach ($this->arrayCharacters as $oCharacters) {
                    $subText = substr($textPara, $oCharacters->pos_start, $oCharacters->pos_len);
                    $subText = str_replace(chr(13), PHP_EOL, $subText);
                    $arrayText = explode(PHP_EOL, $subText);
                    if (end($arrayText) == '') {
                        array_pop($arrayText);
                    }
                    if (reset($arrayText) == '') {
                        array_shift($arrayText);
                    }

                    // Style Character
                    $styleFont = array();
                    if (isset($oCharacters->style)) {
                        if (isset($oCharacters->style->styleFont)) {
                            $styleFont = $oCharacters->style->styleFont;
                        }
                    }

                    foreach ($arrayText as $sText) {
                        // HyperLink
                        if (empty($sText) && !empty($sHYPERLINK)) {
                            $arrHYPERLINK = explode('"', $sHYPERLINK);
                            $oSection->addLink($arrHYPERLINK[1], null);
                            // print_r('>addHyperLink<'.$sHYPERLINK.'>'.ord($sHYPERLINK[0]).EOL);
                            $sHYPERLINK = '';
                        }

                        // TextBreak
                        if (empty($sText)) {
                            $oSection->addTextBreak();
                            $sHYPERLINK = '';
                            // print_r('>addTextBreak<' . EOL);
                        }

                        if (!empty($sText)) {
                            if (!empty($sHYPERLINK) && ord($sText[0]) > 20) {
                                $sHYPERLINK .= $sText;
                            }
                            if (empty($sHYPERLINK)) {
                                if (ord($sText[0]) > 20) {
                                    if (strpos(trim($sText), 'HYPERLINK "') === 0) {
                                        $sHYPERLINK = $sText;
                                    } else {
                                        $oSection->addText($sText, $styleFont);
                                        // print_r('>addText<'.$sText.'>'.ord($sText[0]).EOL);
                                    }
                                }
                                if (ord($sText[0]) == 1) {
                                    if (isset($oCharacters->style->image)) {
                                        $fileImage = tempnam(sys_get_temp_dir(), 'PHPWord_MsDoc') . '.' . $oCharacters->style->image['format'];
                                        file_put_contents($fileImage, $oCharacters->style->image['data']);
                                        $oSection->addImage($fileImage, array('width' => $oCharacters->style->image['width'], 'height' => $oCharacters->style->image['height']));
                                        // print_r('>addImage<'.$fileImage.'>'.EOL);
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    /**
     * Read 8-bit unsigned integer
     *
     * @param string $data
     * @param int $pos
     * @return int
     */
    public static function getInt1d($data, $pos)
    {
        return ord($data[$pos]);
    }

    /**
     * Read 16-bit unsigned integer
     *
     * @param string $data
     * @param int $pos
     * @return int
     */
    public static function getInt2d($data, $pos)
    {
        return ord($data[$pos]) | (ord($data[$pos + 1]) << 8);
    }

    /**
     * Read 24-bit signed integer
     *
     * @param string $data
     * @param int $pos
     * @return int
     */
    public static function getInt3d($data, $pos)
    {
        return ord($data[$pos]) | (ord($data[$pos + 1]) << 8) | (ord($data[$pos + 2]) << 16);
    }

    /**
     * Read 32-bit signed integer
     *
     * @param string $data
     * @param int $pos
     * @return int
     */
    public static function getInt4d($data, $pos)
    {
        // FIX: represent numbers correctly on 64-bit system
        // http://sourceforge.net/tracker/index.php?func=detail&aid=1487372&group_id=99160&atid=623334
        // Hacked by Andreas Rehm 2006 to ensure correct result of the <<24 block on 32 and 64bit systems
        $or24 = ord($data[$pos + 3]);
        if ($or24 >= 128) {
            // negative number
            $ord24 = -abs((256 - $or24) << 24);
        } else {
            $ord24 = ($or24 & 127) << 24;
        }

        return ord($data[$pos]) | (ord($data[$pos + 1]) << 8) | (ord($data[$pos + 2]) << 16) | $ord24;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/ODText/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\ODText;

use PhpOffice\PhpWord\Reader\Word2007\AbstractPart as Word2007AbstractPart;

/**
 * Abstract part reader
 *
 * @since 0.10.0
 * @codeCoverageIgnore
 */
abstract class AbstractPart extends Word2007AbstractPart
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/ODText/Content.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\ODText;

use PhpOffice\PhpWord\Element\TrackChange;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Content reader
 *
 * @since 0.10.0
 */
class Content extends AbstractPart
{
    /**
     * Read content.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);

        $trackedChanges = array();

        $nodes = $xmlReader->getElements('office:body/office:text/*');
        if ($nodes->length > 0) {
            $section = $phpWord->addSection();
            foreach ($nodes as $node) {
                // $styleName = $xmlReader->getAttribute('text:style-name', $node);
                switch ($node->nodeName) {
                    case 'text:h': // Heading
                        $depth = $xmlReader->getAttribute('text:outline-level', $node);
                        $section->addTitle($node->nodeValue, $depth);
                        break;
                    case 'text:p': // Paragraph
                        $children = $node->childNodes;
                        foreach ($children as $child) {
                            switch ($child->nodeName) {
                                case 'text:change-start':
                                    $changeId = $child->getAttribute('text:change-id');
                                    if (isset($trackedChanges[$changeId])) {
                                        $changed = $trackedChanges[$changeId];
                                    }
                                    break;
                                case 'text:change-end':
                                    unset($changed);
                                    break;
                                case 'text:change':
                                    $changeId = $child->getAttribute('text:change-id');
                                    if (isset($trackedChanges[$changeId])) {
                                        $changed = $trackedChanges[$changeId];
                                    }
                                    break;
                            }
                        }

                        $element = $section->addText($node->nodeValue);
                        if (isset($changed) && is_array($changed)) {
                            $element->setTrackChange($changed['changed']);
                            if (isset($changed['textNodes'])) {
                                foreach ($changed['textNodes'] as $changedNode) {
                                    $element = $section->addText($changedNode->nodeValue);
                                    $element->setTrackChange($changed['changed']);
                                }
                            }
                        }
                        break;
                    case 'text:list': // List
                        $listItems = $xmlReader->getElements('text:list-item/text:p', $node);
                        foreach ($listItems as $listItem) {
                            // $listStyleName = $xmlReader->getAttribute('text:style-name', $listItem);
                            $section->addListItem($listItem->nodeValue, 0);
                        }
                        break;
                    case 'text:tracked-changes':
                        $changedRegions = $xmlReader->getElements('text:changed-region', $node);
                        foreach ($changedRegions as $changedRegion) {
                            $type = ($changedRegion->firstChild->nodeName == 'text:insertion') ? TrackChange::INSERTED : TrackChange::DELETED;
                            $creatorNode = $xmlReader->getElements('office:change-info/dc:creator', $changedRegion->firstChild);
                            $author = $creatorNode[0]->nodeValue;
                            $dateNode = $xmlReader->getElements('office:change-info/dc:date', $changedRegion->firstChild);
                            $date = $dateNode[0]->nodeValue;
                            $date = preg_replace('/\.\d+$/', '', $date);
                            $date = \DateTime::createFromFormat('Y-m-d\TH:i:s', $date);
                            $changed = new TrackChange($type, $author, $date);
                            $textNodes = $xmlReader->getElements('text:deletion/text:p', $changedRegion);
                            $trackedChanges[$changedRegion->getAttribute('text:id')] = array('changed'  => $changed, 'textNodes'=> $textNodes);
                        }
                        break;
                }
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/ODText/Meta.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\ODText;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Meta reader
 *
 * @since 0.11.0
 */
class Meta extends AbstractPart
{
    /**
     * Read meta.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     * @todo Process property type
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);
        $docProps = $phpWord->getDocInfo();

        $metaNode = $xmlReader->getElement('office:meta');

        // Standard properties
        $properties = array(
            'title'          => 'dc:title',
            'subject'        => 'dc:subject',
            'description'    => 'dc:description',
            'keywords'       => 'meta:keyword',
            'creator'        => 'meta:initial-creator',
            'lastModifiedBy' => 'dc:creator',
            // 'created'        => 'meta:creation-date',
            // 'modified'       => 'dc:date',
        );
        foreach ($properties as $property => $path) {
            $method = "set{$property}";
            $propertyNode = $xmlReader->getElement($path, $metaNode);
            if ($propertyNode !== null && method_exists($docProps, $method)) {
                $docProps->$method($propertyNode->nodeValue);
            }
        }

        // Custom properties
        $propertyNodes = $xmlReader->getElements('meta:user-defined', $metaNode);
        foreach ($propertyNodes as $propertyNode) {
            $property = $xmlReader->getAttribute('meta:name', $propertyNode);

            // Set category, company, and manager property
            if (in_array($property, array('Category', 'Company', 'Manager'))) {
                $method = "set{$property}";
                $docProps->$method($propertyNode->nodeValue);
            } else {
                // Set other custom properties
                $docProps->setCustomProperty($property, $propertyNode->nodeValue);
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/ODText.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Reader for ODText
 *
 * @since 0.10.0
 */
class ODText extends AbstractReader implements ReaderInterface
{
    /**
     * Loads PhpWord from file
     *
     * @param string $docFile
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function load($docFile)
    {
        $phpWord = new PhpWord();
        $relationships = $this->readRelationships($docFile);

        $readerParts = array(
            'content.xml' => 'Content',
            'meta.xml'    => 'Meta',
        );

        foreach ($readerParts as $xmlFile => $partName) {
            $this->readPart($phpWord, $relationships, $partName, $docFile, $xmlFile);
        }

        return $phpWord;
    }

    /**
     * Read document part.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     * @param array $relationships
     * @param string $partName
     * @param string $docFile
     * @param string $xmlFile
     */
    private function readPart(PhpWord $phpWord, $relationships, $partName, $docFile, $xmlFile)
    {
        $partClass = "PhpOffice\\PhpWord\\Reader\\ODText\\{$partName}";
        if (class_exists($partClass)) {
            /** @var \PhpOffice\PhpWord\Reader\ODText\AbstractPart $part Type hint */
            $part = new $partClass($docFile, $xmlFile);
            $part->setRels($relationships);
            $part->read($phpWord);
        }
    }

    /**
     * Read all relationship files
     *
     * @param string $docFile
     * @return array
     */
    private function readRelationships($docFile)
    {
        $rels = array();
        $xmlFile = 'META-INF/manifest.xml';
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($docFile, $xmlFile);
        $nodes = $xmlReader->getElements('manifest:file-entry');
        foreach ($nodes as $node) {
            $type = $xmlReader->getAttribute('manifest:media-type', $node);
            $target = $xmlReader->getAttribute('manifest:full-path', $node);
            $rels[] = array('type' => $type, 'target' => $target);
        }

        return $rels;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/ReaderInterface.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

/**
 * Reader interface
 *
 * @since 0.8.0
 */
interface ReaderInterface
{
    /**
     * Can the current ReaderInterface read the file?
     *
     * @param  string $filename
     * @return bool
     */
    public function canRead($filename);

    /**
     * Loads PhpWord from file
     *
     * @param string $filename
     */
    public function load($filename);
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/RTF/Document.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\RTF;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\SimpleType\Jc;

/**
 * RTF document reader
 *
 * References:
 * - How to Write an RTF Reader http://latex2rtf.sourceforge.net/rtfspec_45.html
 * - PHP rtfclass by Markus Fischer https://github.com/mfn/rtfclass
 * - JavaScript RTF-parser by LazyGyu https://github.com/lazygyu/RTF-parser
 *
 * @since 0.11.0
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod)
 */
class Document
{
    /** @const int */
    const PARA = 'readParagraph';
    const STYL = 'readStyle';
    const SKIP = 'readSkip';

    /**
     * PhpWord object
     *
     * @var \PhpOffice\PhpWord\PhpWord
     */
    private $phpWord;

    /**
     * Section object
     *
     * @var \PhpOffice\PhpWord\Element\Section
     */
    private $section;

    /**
     * Textrun object
     *
     * @var \PhpOffice\PhpWord\Element\TextRun
     */
    private $textrun;

    /**
     * RTF content
     *
     * @var string
     */
    public $rtf;

    /**
     * Content length
     *
     * @var int
     */
    private $length = 0;

    /**
     * Character index
     *
     * @var int
     */
    private $offset = 0;

    /**
     * Current control word
     *
     * @var string
     */
    private $control = '';

    /**
     * Text content
     *
     * @var string
     */
    private $text = '';

    /**
     * Parsing a control word flag
     *
     * @var bool
     */
    private $isControl = false;

    /**
     * First character flag: watch out for control symbols
     *
     * @var bool
     */
    private $isFirst = false;

    /**
     * Group groups
     *
     * @var array
     */
    private $groups = array();

    /**
     * Parser flags; not used
     *
     * @var array
     */
    private $flags = array();

    /**
     * Parse RTF content
     *
     * - Marks controlling characters `{`, `}`, and `\`
     * - Removes line endings
     * - Builds control words and control symbols
     * - Pushes every other character into the text queue
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     * @todo Use `fread` stream for scalability
     */
    public function read(PhpWord $phpWord)
    {
        $markers = array(
            123 => 'markOpening',   // {
            125 => 'markClosing',   // }
            92  => 'markBackslash', // \
            10  => 'markNewline',   // LF
            13  => 'markNewline',   // CR
        );

        $this->phpWord = $phpWord;
        $this->section = $phpWord->addSection();
        $this->textrun = $this->section->addTextRun();
        $this->length = strlen($this->rtf);

        $this->flags['paragraph'] = true; // Set paragraph flag from the beginning

        // Walk each characters
        while ($this->offset < $this->length) {
            $char = $this->rtf[$this->offset];
            $ascii = ord($char);

            if (isset($markers[$ascii])) { // Marker found: {, }, \, LF, or CR
                $markerFunction = $markers[$ascii];
                $this->$markerFunction();
            } else {
                if (false === $this->isControl) { // Non control word: Push character
                    $this->pushText($char);
                } else {
                    if (preg_match('/^[a-zA-Z0-9-]?$/', $char)) { // No delimiter: Buffer control
                        $this->control .= $char;
                        $this->isFirst = false;
                    } else { // Delimiter found: Parse buffered control
                        if ($this->isFirst) {
                            $this->isFirst = false;
                        } else {
                            if (' ' == $char) { // Discard space as a control word delimiter
                                $this->flushControl(true);
                            }
                        }
                    }
                }
            }
            $this->offset++;
        }
        $this->flushText();
    }

    /**
     * Mark opening braket `{` character.
     */
    private function markOpening()
    {
        $this->flush(true);
        array_push($this->groups, $this->flags);
    }

    /**
     * Mark closing braket `}` character.
     */
    private function markClosing()
    {
        $this->flush(true);
        $this->flags = array_pop($this->groups);
    }

    /**
     * Mark backslash `\` character.
     */
    private function markBackslash()
    {
        if ($this->isFirst) {
            $this->setControl(false);
            $this->text .= '\\';
        } else {
            $this->flush();
            $this->setControl(true);
            $this->control = '';
        }
    }

    /**
     * Mark newline character: Flush control word because it's not possible to span multiline.
     */
    private function markNewline()
    {
        if ($this->isControl) {
            $this->flushControl(true);
        }
    }

    /**
     * Flush control word or text.
     *
     * @param bool $isControl
     */
    private function flush($isControl = false)
    {
        if ($this->isControl) {
            $this->flushControl($isControl);
        } else {
            $this->flushText();
        }
    }

    /**
     * Flush control word.
     *
     * @param bool $isControl
     */
    private function flushControl($isControl = false)
    {
        if (1 === preg_match('/^([A-Za-z]+)(-?[0-9]*) ?$/', $this->control, $match)) {
            list(, $control, $parameter) = $match;
            $this->parseControl($control, $parameter);
        }

        if (true === $isControl) {
            $this->setControl(false);
        }
    }

    /**
     * Flush text in queue.
     */
    private function flushText()
    {
        if ($this->text != '') {
            if (isset($this->flags['property'])) { // Set property
                $this->flags['value'] = $this->text;
            } else { // Set text
                if (true === $this->flags['paragraph']) {
                    $this->flags['paragraph'] = false;
                    $this->flags['text'] = $this->text;
                }
            }

            // Add text if it's not flagged as skipped
            if (!isset($this->flags['skipped'])) {
                $this->readText();
            }

            $this->text = '';
        }
    }

    /**
     * Reset control word and first char state.
     *
     * @param bool $value
     */
    private function setControl($value)
    {
        $this->isControl = $value;
        $this->isFirst = $value;
    }

    /**
     * Push text into queue.
     *
     * @param string $char
     */
    private function pushText($char)
    {
        if ('<' == $char) {
            $this->text .= '&lt;';
        } elseif ('>' == $char) {
            $this->text .= '&gt;';
        } else {
            $this->text .= $char;
        }
    }

    /**
     * Parse control.
     *
     * @param string $control
     * @param string $parameter
     */
    private function parseControl($control, $parameter)
    {
        $controls = array(
            'par'       => array(self::PARA,    'paragraph',    true),
            'b'         => array(self::STYL,    'font',         'bold',          true),
            'i'         => array(self::STYL,    'font',         'italic',        true),
            'u'         => array(self::STYL,    'font',         'underline',     true),
            'strike'    => array(self::STYL,    'font',         'strikethrough', true),
            'fs'        => array(self::STYL,    'font',         'size',          $parameter),
            'qc'        => array(self::STYL,    'paragraph',    'alignment',     Jc::CENTER),
            'sa'        => array(self::STYL,    'paragraph',    'spaceAfter',    $parameter),
            'fonttbl'   => array(self::SKIP,    'fonttbl',      null),
            'colortbl'  => array(self::SKIP,    'colortbl',     null),
            'info'      => array(self::SKIP,    'info',         null),
            'generator' => array(self::SKIP,    'generator',    null),
            'title'     => array(self::SKIP,    'title',        null),
            'subject'   => array(self::SKIP,    'subject',      null),
            'category'  => array(self::SKIP,    'category',     null),
            'keywords'  => array(self::SKIP,    'keywords',     null),
            'comment'   => array(self::SKIP,    'comment',      null),
            'shppict'   => array(self::SKIP,    'pic',          null),
            'fldinst'   => array(self::SKIP,    'link',         null),
        );

        if (isset($controls[$control])) {
            list($function) = $controls[$control];
            if (method_exists($this, $function)) {
                $directives = $controls[$control];
                array_shift($directives); // remove the function variable; we won't need it
                $this->$function($directives);
            }
        }
    }

    /**
     * Read paragraph.
     *
     * @param array $directives
     */
    private function readParagraph($directives)
    {
        list($property, $value) = $directives;
        $this->textrun = $this->section->addTextRun();
        $this->flags[$property] = $value;
    }

    /**
     * Read style.
     *
     * @param array $directives
     */
    private function readStyle($directives)
    {
        list($style, $property, $value) = $directives;
        $this->flags['styles'][$style][$property] = $value;
    }

    /**
     * Read skip.
     *
     * @param array $directives
     */
    private function readSkip($directives)
    {
        list($property) = $directives;
        $this->flags['property'] = $property;
        $this->flags['skipped'] = true;
    }

    /**
     * Read text.
     */
    private function readText()
    {
        $text = $this->textrun->addText($this->text);
        if (isset($this->flags['styles']['font'])) {
            $text->getFontStyle()->setStyleByArray($this->flags['styles']['font']);
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/RTF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Reader\RTF\Document;

/**
 * RTF Reader class
 *
 * @since 0.11.0
 */
class RTF extends AbstractReader implements ReaderInterface
{
    /**
     * Loads PhpWord from file
     *
     * @param string $docFile
     *
     * @throws \Exception
     *
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function load($docFile)
    {
        $phpWord = new PhpWord();

        if ($this->canRead($docFile)) {
            $doc = new Document();
            $doc->rtf = file_get_contents($docFile);
            $doc->read($phpWord);
        } else {
            throw new \Exception("Cannot read {$docFile}.");
        }

        return $phpWord;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\ComplexType\TblWidth as TblWidthComplexType;
use PhpOffice\PhpWord\Element\AbstractContainer;
use PhpOffice\PhpWord\Element\TextRun;
use PhpOffice\PhpWord\Element\TrackChange;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Abstract part reader
 *
 * This class is inherited by ODText reader
 *
 * @since 0.10.0
 */
abstract class AbstractPart
{
    /**
     * Conversion method
     *
     * @const int
     */
    const READ_VALUE = 'attributeValue';            // Read attribute value
    const READ_EQUAL = 'attributeEquals';           // Read `true` when attribute value equals specified value
    const READ_TRUE = 'attributeTrue';              // Read `true` when element exists
    const READ_FALSE = 'attributeFalse';            // Read `false` when element exists
    const READ_SIZE = 'attributeMultiplyByTwo';     // Read special attribute value for Font::$size

    /**
     * Document file
     *
     * @var string
     */
    protected $docFile;

    /**
     * XML file
     *
     * @var string
     */
    protected $xmlFile;

    /**
     * Part relationships
     *
     * @var array
     */
    protected $rels = array();

    /**
     * Read part.
     */
    abstract public function read(PhpWord $phpWord);

    /**
     * Create new instance
     *
     * @param string $docFile
     * @param string $xmlFile
     */
    public function __construct($docFile, $xmlFile)
    {
        $this->docFile = $docFile;
        $this->xmlFile = $xmlFile;
    }

    /**
     * Set relationships.
     *
     * @param array $value
     */
    public function setRels($value)
    {
        $this->rels = $value;
    }

    /**
     * Read w:p.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @param \PhpOffice\PhpWord\Element\AbstractContainer $parent
     * @param string $docPart
     *
     * @todo Get font style for preserve text
     */
    protected function readParagraph(XMLReader $xmlReader, \DOMElement $domNode, $parent, $docPart = 'document')
    {
        // Paragraph style
        $paragraphStyle = null;
        $headingDepth = null;
        if ($xmlReader->elementExists('w:pPr', $domNode)) {
            $paragraphStyle = $this->readParagraphStyle($xmlReader, $domNode);
            $headingDepth = $this->getHeadingDepth($paragraphStyle);
        }

        // PreserveText
        if ($xmlReader->elementExists('w:r/w:instrText', $domNode)) {
            $ignoreText = false;
            $textContent = '';
            $fontStyle = $this->readFontStyle($xmlReader, $domNode);
            $nodes = $xmlReader->getElements('w:r', $domNode);
            foreach ($nodes as $node) {
                $instrText = $xmlReader->getValue('w:instrText', $node);
                if ($xmlReader->elementExists('w:fldChar', $node)) {
                    $fldCharType = $xmlReader->getAttribute('w:fldCharType', $node, 'w:fldChar');
                    if ('begin' == $fldCharType) {
                        $ignoreText = true;
                    } elseif ('end' == $fldCharType) {
                        $ignoreText = false;
                    }
                }
                if (!is_null($instrText)) {
                    $textContent .= '{' . $instrText . '}';
                } else {
                    if (false === $ignoreText) {
                        $textContent .= $xmlReader->getValue('w:t', $node);
                    }
                }
            }
            $parent->addPreserveText(htmlspecialchars($textContent, ENT_QUOTES, 'UTF-8'), $fontStyle, $paragraphStyle);
        } elseif ($xmlReader->elementExists('w:pPr/w:numPr', $domNode)) {
            // List item
            $numId = $xmlReader->getAttribute('w:val', $domNode, 'w:pPr/w:numPr/w:numId');
            $levelId = $xmlReader->getAttribute('w:val', $domNode, 'w:pPr/w:numPr/w:ilvl');
            $nodes = $xmlReader->getElements('*', $domNode);

            $listItemRun = $parent->addListItemRun($levelId, "PHPWordList{$numId}", $paragraphStyle);

            foreach ($nodes as $node) {
                $this->readRun($xmlReader, $node, $listItemRun, $docPart, $paragraphStyle);
            }
        } elseif ($headingDepth !== null) {
            // Heading or Title
            $textContent = null;
            $nodes = $xmlReader->getElements('w:r', $domNode);
            if ($nodes->length === 1) {
                $textContent = htmlspecialchars($xmlReader->getValue('w:t', $nodes->item(0)), ENT_QUOTES, 'UTF-8');
            } else {
                $textContent = new TextRun($paragraphStyle);
                foreach ($nodes as $node) {
                    $this->readRun($xmlReader, $node, $textContent, $docPart, $paragraphStyle);
                }
            }
            $parent->addTitle($textContent, $headingDepth);
        } else {
            // Text and TextRun
            $textRunContainers = $xmlReader->countElements('w:r|w:ins|w:del|w:hyperlink|w:smartTag', $domNode);
            if (0 === $textRunContainers) {
                $parent->addTextBreak(null, $paragraphStyle);
            } else {
                $nodes = $xmlReader->getElements('*', $domNode);
                $paragraph = $parent->addTextRun($paragraphStyle);
                foreach ($nodes as $node) {
                    $this->readRun($xmlReader, $node, $paragraph, $docPart, $paragraphStyle);
                }
            }
        }
    }

    /**
     * Returns the depth of the Heading, returns 0 for a Title
     *
     * @param array $paragraphStyle
     * @return number|null
     */
    private function getHeadingDepth(array $paragraphStyle = null)
    {
        if (is_array($paragraphStyle) && isset($paragraphStyle['styleName'])) {
            if ('Title' === $paragraphStyle['styleName']) {
                return 0;
            }

            $headingMatches = array();
            preg_match('/Heading(\d)/', $paragraphStyle['styleName'], $headingMatches);
            if (!empty($headingMatches)) {
                return $headingMatches[1];
            }
        }

        return null;
    }

    /**
     * Read w:r.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @param \PhpOffice\PhpWord\Element\AbstractContainer $parent
     * @param string $docPart
     * @param mixed $paragraphStyle
     *
     * @todo Footnote paragraph style
     */
    protected function readRun(XMLReader $xmlReader, \DOMElement $domNode, $parent, $docPart, $paragraphStyle = null)
    {
        if (in_array($domNode->nodeName, array('w:ins', 'w:del', 'w:smartTag', 'w:hyperlink'))) {
            $nodes = $xmlReader->getElements('*', $domNode);
            foreach ($nodes as $node) {
                $this->readRun($xmlReader, $node, $parent, $docPart, $paragraphStyle);
            }
        } elseif ($domNode->nodeName == 'w:r') {
            $fontStyle = $this->readFontStyle($xmlReader, $domNode);
            $nodes = $xmlReader->getElements('*', $domNode);
            foreach ($nodes as $node) {
                $this->readRunChild($xmlReader, $node, $parent, $docPart, $paragraphStyle, $fontStyle);
            }
        }
    }

    /**
     * Parses nodes under w:r
     *
     * @param XMLReader $xmlReader
     * @param \DOMElement $node
     * @param AbstractContainer $parent
     * @param string $docPart
     * @param mixed $paragraphStyle
     * @param mixed $fontStyle
     */
    protected function readRunChild(XMLReader $xmlReader, \DOMElement $node, AbstractContainer $parent, $docPart, $paragraphStyle = null, $fontStyle = null)
    {
        $runParent = $node->parentNode->parentNode;
        if ($node->nodeName == 'w:footnoteReference') {
            // Footnote
            $wId = $xmlReader->getAttribute('w:id', $node);
            $footnote = $parent->addFootnote();
            $footnote->setRelationId($wId);
        } elseif ($node->nodeName == 'w:endnoteReference') {
            // Endnote
            $wId = $xmlReader->getAttribute('w:id', $node);
            $endnote = $parent->addEndnote();
            $endnote->setRelationId($wId);
        } elseif ($node->nodeName == 'w:pict') {
            // Image
            $rId = $xmlReader->getAttribute('r:id', $node, 'v:shape/v:imagedata');
            $target = $this->getMediaTarget($docPart, $rId);
            if (!is_null($target)) {
                if ('External' == $this->getTargetMode($docPart, $rId)) {
                    $imageSource = $target;
                } else {
                    $imageSource = "zip://{$this->docFile}#{$target}";
                }
                $parent->addImage($imageSource);
            }
        } elseif ($node->nodeName == 'w:drawing') {
            // Office 2011 Image
            $xmlReader->registerNamespace('wp', 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing');
            $xmlReader->registerNamespace('r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
            $xmlReader->registerNamespace('pic', 'http://schemas.openxmlformats.org/drawingml/2006/picture');
            $xmlReader->registerNamespace('a', 'http://schemas.openxmlformats.org/drawingml/2006/main');

            $name = $xmlReader->getAttribute('name', $node, 'wp:inline/a:graphic/a:graphicData/pic:pic/pic:nvPicPr/pic:cNvPr');
            $embedId = $xmlReader->getAttribute('r:embed', $node, 'wp:inline/a:graphic/a:graphicData/pic:pic/pic:blipFill/a:blip');
            if ($name === null && $embedId === null) { // some Converters puts images on a different path
                $name = $xmlReader->getAttribute('name', $node, 'wp:anchor/a:graphic/a:graphicData/pic:pic/pic:nvPicPr/pic:cNvPr');
                $embedId = $xmlReader->getAttribute('r:embed', $node, 'wp:anchor/a:graphic/a:graphicData/pic:pic/pic:blipFill/a:blip');
            }
            $target = $this->getMediaTarget($docPart, $embedId);
            if (!is_null($target)) {
                $imageSource = "zip://{$this->docFile}#{$target}";
                $parent->addImage($imageSource, null, false, $name);
            }
        } elseif ($node->nodeName == 'w:object') {
            // Object
            $rId = $xmlReader->getAttribute('r:id', $node, 'o:OLEObject');
            // $rIdIcon = $xmlReader->getAttribute('r:id', $domNode, 'w:object/v:shape/v:imagedata');
            $target = $this->getMediaTarget($docPart, $rId);
            if (!is_null($target)) {
                $textContent = "&lt;Object: {$target}>";
                $parent->addText($textContent, $fontStyle, $paragraphStyle);
            }
        } elseif ($node->nodeName == 'w:br') {
            $parent->addTextBreak();
        } elseif ($node->nodeName == 'w:tab') {
            $parent->addText("\t");
        } elseif ($node->nodeName == 'mc:AlternateContent') {
            if ($node->hasChildNodes()) {
                // Get fallback instead of mc:Choice to make sure it is compatible
                $fallbackElements = $node->getElementsByTagName('Fallback');

                if ($fallbackElements->length) {
                    $fallback = $fallbackElements->item(0);
                    // TextRun
                    $textContent = htmlspecialchars($fallback->nodeValue, ENT_QUOTES, 'UTF-8');

                    $parent->addText($textContent, $fontStyle, $paragraphStyle);
                }
            }
        } elseif ($node->nodeName == 'w:t' || $node->nodeName == 'w:delText') {
            // TextRun
            $textContent = htmlspecialchars($xmlReader->getValue('.', $node), ENT_QUOTES, 'UTF-8');

            if ($runParent->nodeName == 'w:hyperlink') {
                $rId = $xmlReader->getAttribute('r:id', $runParent);
                $target = $this->getMediaTarget($docPart, $rId);
                if (!is_null($target)) {
                    $parent->addLink($target, $textContent, $fontStyle, $paragraphStyle);
                } else {
                    $parent->addText($textContent, $fontStyle, $paragraphStyle);
                }
            } else {
                /** @var AbstractElement $element */
                $element = $parent->addText($textContent, $fontStyle, $paragraphStyle);
                if (in_array($runParent->nodeName, array('w:ins', 'w:del'))) {
                    $type = ($runParent->nodeName == 'w:del') ? TrackChange::DELETED : TrackChange::INSERTED;
                    $author = $runParent->getAttribute('w:author');
                    $date = \DateTime::createFromFormat('Y-m-d\TH:i:s\Z', $runParent->getAttribute('w:date'));
                    $element->setChangeInfo($type, $author, $date);
                }
            }
        }
    }

    /**
     * Read w:tbl.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @param mixed $parent
     * @param string $docPart
     */
    protected function readTable(XMLReader $xmlReader, \DOMElement $domNode, $parent, $docPart = 'document')
    {
        // Table style
        $tblStyle = null;
        if ($xmlReader->elementExists('w:tblPr', $domNode)) {
            $tblStyle = $this->readTableStyle($xmlReader, $domNode);
        }

        /** @var \PhpOffice\PhpWord\Element\Table $table Type hint */
        $table = $parent->addTable($tblStyle);
        $tblNodes = $xmlReader->getElements('*', $domNode);
        foreach ($tblNodes as $tblNode) {
            if ('w:tblGrid' == $tblNode->nodeName) { // Column
                // @todo Do something with table columns
            } elseif ('w:tr' == $tblNode->nodeName) { // Row
                $rowHeight = $xmlReader->getAttribute('w:val', $tblNode, 'w:trPr/w:trHeight');
                $rowHRule = $xmlReader->getAttribute('w:hRule', $tblNode, 'w:trPr/w:trHeight');
                $rowHRule = $rowHRule == 'exact';
                $rowStyle = array(
                    'tblHeader'   => $xmlReader->elementExists('w:trPr/w:tblHeader', $tblNode),
                    'cantSplit'   => $xmlReader->elementExists('w:trPr/w:cantSplit', $tblNode),
                    'exactHeight' => $rowHRule,
                );

                $row = $table->addRow($rowHeight, $rowStyle);
                $rowNodes = $xmlReader->getElements('*', $tblNode);
                foreach ($rowNodes as $rowNode) {
                    if ('w:trPr' == $rowNode->nodeName) { // Row style
                        // @todo Do something with row style
                    } elseif ('w:tc' == $rowNode->nodeName) { // Cell
                        $cellWidth = $xmlReader->getAttribute('w:w', $rowNode, 'w:tcPr/w:tcW');
                        $cellStyle = null;
                        $cellStyleNode = $xmlReader->getElement('w:tcPr', $rowNode);
                        if (!is_null($cellStyleNode)) {
                            $cellStyle = $this->readCellStyle($xmlReader, $cellStyleNode);
                        }

                        $cell = $row->addCell($cellWidth, $cellStyle);
                        $cellNodes = $xmlReader->getElements('*', $rowNode);
                        foreach ($cellNodes as $cellNode) {
                            if ('w:p' == $cellNode->nodeName) { // Paragraph
                                $this->readParagraph($xmlReader, $cellNode, $cell, $docPart);
                            }
                        }
                    }
                }
            }
        }
    }

    /**
     * Read w:pPr.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return array|null
     */
    protected function readParagraphStyle(XMLReader $xmlReader, \DOMElement $domNode)
    {
        if (!$xmlReader->elementExists('w:pPr', $domNode)) {
            return null;
        }

        $styleNode = $xmlReader->getElement('w:pPr', $domNode);
        $styleDefs = array(
            'styleName'           => array(self::READ_VALUE, array('w:pStyle', 'w:name')),
            'alignment'           => array(self::READ_VALUE, 'w:jc'),
            'basedOn'             => array(self::READ_VALUE, 'w:basedOn'),
            'next'                => array(self::READ_VALUE, 'w:next'),
            'indent'              => array(self::READ_VALUE, 'w:ind', 'w:left'),
            'hanging'             => array(self::READ_VALUE, 'w:ind', 'w:hanging'),
            'spaceAfter'          => array(self::READ_VALUE, 'w:spacing', 'w:after'),
            'spaceBefore'         => array(self::READ_VALUE, 'w:spacing', 'w:before'),
            'widowControl'        => array(self::READ_FALSE, 'w:widowControl'),
            'keepNext'            => array(self::READ_TRUE,  'w:keepNext'),
            'keepLines'           => array(self::READ_TRUE,  'w:keepLines'),
            'pageBreakBefore'     => array(self::READ_TRUE,  'w:pageBreakBefore'),
            'contextualSpacing'   => array(self::READ_TRUE,  'w:contextualSpacing'),
            'bidi'                => array(self::READ_TRUE,  'w:bidi'),
            'suppressAutoHyphens' => array(self::READ_TRUE,  'w:suppressAutoHyphens'),
        );

        return $this->readStyleDefs($xmlReader, $styleNode, $styleDefs);
    }

    /**
     * Read w:rPr
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return array|null
     */
    protected function readFontStyle(XMLReader $xmlReader, \DOMElement $domNode)
    {
        if (is_null($domNode)) {
            return null;
        }
        // Hyperlink has an extra w:r child
        if ('w:hyperlink' == $domNode->nodeName) {
            $domNode = $xmlReader->getElement('w:r', $domNode);
        }
        if (!$xmlReader->elementExists('w:rPr', $domNode)) {
            return null;
        }

        $styleNode = $xmlReader->getElement('w:rPr', $domNode);
        $styleDefs = array(
            'styleName'           => array(self::READ_VALUE, 'w:rStyle'),
            'name'                => array(self::READ_VALUE, 'w:rFonts', array('w:ascii', 'w:hAnsi', 'w:eastAsia', 'w:cs')),
            'hint'                => array(self::READ_VALUE, 'w:rFonts', 'w:hint'),
            'size'                => array(self::READ_SIZE,  array('w:sz', 'w:szCs')),
            'color'               => array(self::READ_VALUE, 'w:color'),
            'underline'           => array(self::READ_VALUE, 'w:u'),
            'bold'                => array(self::READ_TRUE,  'w:b'),
            'italic'              => array(self::READ_TRUE,  'w:i'),
            'strikethrough'       => array(self::READ_TRUE,  'w:strike'),
            'doubleStrikethrough' => array(self::READ_TRUE,  'w:dstrike'),
            'smallCaps'           => array(self::READ_TRUE,  'w:smallCaps'),
            'allCaps'             => array(self::READ_TRUE,  'w:caps'),
            'superScript'         => array(self::READ_EQUAL, 'w:vertAlign', 'w:val', 'superscript'),
            'subScript'           => array(self::READ_EQUAL, 'w:vertAlign', 'w:val', 'subscript'),
            'fgColor'             => array(self::READ_VALUE, 'w:highlight'),
            'rtl'                 => array(self::READ_TRUE,  'w:rtl'),
            'lang'                => array(self::READ_VALUE, 'w:lang'),
            'position'            => array(self::READ_VALUE, 'w:position'),
            'hidden'              => array(self::READ_TRUE,  'w:vanish'),
        );

        return $this->readStyleDefs($xmlReader, $styleNode, $styleDefs);
    }

    /**
     * Read w:tblPr
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return string|array|null
     * @todo Capture w:tblStylePr w:type="firstRow"
     */
    protected function readTableStyle(XMLReader $xmlReader, \DOMElement $domNode)
    {
        $style = null;
        $margins = array('top', 'left', 'bottom', 'right');
        $borders = array_merge($margins, array('insideH', 'insideV'));

        if ($xmlReader->elementExists('w:tblPr', $domNode)) {
            if ($xmlReader->elementExists('w:tblPr/w:tblStyle', $domNode)) {
                $style = $xmlReader->getAttribute('w:val', $domNode, 'w:tblPr/w:tblStyle');
            } else {
                $styleNode = $xmlReader->getElement('w:tblPr', $domNode);
                $styleDefs = array();
                foreach ($margins as $side) {
                    $ucfSide = ucfirst($side);
                    $styleDefs["cellMargin$ucfSide"] = array(self::READ_VALUE, "w:tblCellMar/w:$side", 'w:w');
                }
                foreach ($borders as $side) {
                    $ucfSide = ucfirst($side);
                    $styleDefs["border{$ucfSide}Size"] = array(self::READ_VALUE, "w:tblBorders/w:$side", 'w:sz');
                    $styleDefs["border{$ucfSide}Color"] = array(self::READ_VALUE, "w:tblBorders/w:$side", 'w:color');
                    $styleDefs["border{$ucfSide}Style"] = array(self::READ_VALUE, "w:tblBorders/w:$side", 'w:val');
                }
                $styleDefs['layout'] = array(self::READ_VALUE, 'w:tblLayout', 'w:type');
                $styleDefs['bidiVisual'] = array(self::READ_TRUE, 'w:bidiVisual');
                $styleDefs['cellSpacing'] = array(self::READ_VALUE, 'w:tblCellSpacing', 'w:w');
                $style = $this->readStyleDefs($xmlReader, $styleNode, $styleDefs);

                $tablePositionNode = $xmlReader->getElement('w:tblpPr', $styleNode);
                if ($tablePositionNode !== null) {
                    $style['position'] = $this->readTablePosition($xmlReader, $tablePositionNode);
                }

                $indentNode = $xmlReader->getElement('w:tblInd', $styleNode);
                if ($indentNode !== null) {
                    $style['indent'] = $this->readTableIndent($xmlReader, $indentNode);
                }
            }
        }

        return $style;
    }

    /**
     * Read w:tblpPr
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return array
     */
    private function readTablePosition(XMLReader $xmlReader, \DOMElement $domNode)
    {
        $styleDefs = array(
            'leftFromText'   => array(self::READ_VALUE, '.', 'w:leftFromText'),
            'rightFromText'  => array(self::READ_VALUE, '.', 'w:rightFromText'),
            'topFromText'    => array(self::READ_VALUE, '.', 'w:topFromText'),
            'bottomFromText' => array(self::READ_VALUE, '.', 'w:bottomFromText'),
            'vertAnchor'     => array(self::READ_VALUE, '.', 'w:vertAnchor'),
            'horzAnchor'     => array(self::READ_VALUE, '.', 'w:horzAnchor'),
            'tblpXSpec'      => array(self::READ_VALUE, '.', 'w:tblpXSpec'),
            'tblpX'          => array(self::READ_VALUE, '.', 'w:tblpX'),
            'tblpYSpec'      => array(self::READ_VALUE, '.', 'w:tblpYSpec'),
            'tblpY'          => array(self::READ_VALUE, '.', 'w:tblpY'),
        );

        return $this->readStyleDefs($xmlReader, $domNode, $styleDefs);
    }

    /**
     * Read w:tblInd
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return TblWidthComplexType
     */
    private function readTableIndent(XMLReader $xmlReader, \DOMElement $domNode)
    {
        $styleDefs = array(
            'value' => array(self::READ_VALUE, '.', 'w:w'),
            'type'  => array(self::READ_VALUE, '.', 'w:type'),
        );
        $styleDefs = $this->readStyleDefs($xmlReader, $domNode, $styleDefs);

        return new TblWidthComplexType((int) $styleDefs['value'], $styleDefs['type']);
    }

    /**
     * Read w:tcPr
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @return array
     */
    private function readCellStyle(XMLReader $xmlReader, \DOMElement $domNode)
    {
        $styleDefs = array(
            'valign'        => array(self::READ_VALUE, 'w:vAlign'),
            'textDirection' => array(self::READ_VALUE, 'w:textDirection'),
            'gridSpan'      => array(self::READ_VALUE, 'w:gridSpan'),
            'vMerge'        => array(self::READ_VALUE, 'w:vMerge'),
            'bgColor'       => array(self::READ_VALUE, 'w:shd', 'w:fill'),
        );

        return $this->readStyleDefs($xmlReader, $domNode, $styleDefs);
    }

    /**
     * Returns the first child element found
     *
     * @param XMLReader $xmlReader
     * @param \DOMElement|null $parentNode
     * @param string|array|null $elements
     * @return string|null
     */
    private function findPossibleElement(XMLReader $xmlReader, \DOMElement $parentNode = null, $elements = null)
    {
        if (is_array($elements)) {
            //if element is an array, we take the first element that exists in the XML
            foreach ($elements as $possibleElement) {
                if ($xmlReader->elementExists($possibleElement, $parentNode)) {
                    return $possibleElement;
                }
            }
        } else {
            return $elements;
        }

        return null;
    }

    /**
     * Returns the first attribute found
     *
     * @param XMLReader $xmlReader
     * @param \DOMElement $node
     * @param string|array $attributes
     * @return string|null
     */
    private function findPossibleAttribute(XMLReader $xmlReader, \DOMElement $node, $attributes)
    {
        //if attribute is an array, we take the first attribute that exists in the XML
        if (is_array($attributes)) {
            foreach ($attributes as $possibleAttribute) {
                if ($xmlReader->getAttribute($possibleAttribute, $node)) {
                    return $possibleAttribute;
                }
            }

            return null;
        }

        return $attributes;
    }

    /**
     * Read style definition
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $parentNode
     * @param array $styleDefs
     * @ignoreScrutinizerPatch
     * @return array
     */
    protected function readStyleDefs(XMLReader $xmlReader, \DOMElement $parentNode = null, $styleDefs = array())
    {
        $styles = array();

        foreach ($styleDefs as $styleProp => $styleVal) {
            list($method, $element, $attribute, $expected) = array_pad($styleVal, 4, null);

            $element = $this->findPossibleElement($xmlReader, $parentNode, $element);
            if ($element === null) {
                continue;
            }

            if ($xmlReader->elementExists($element, $parentNode)) {
                $node = $xmlReader->getElement($element, $parentNode);

                $attribute = $this->findPossibleAttribute($xmlReader, $node, $attribute);

                // Use w:val as default if no attribute assigned
                $attribute = ($attribute === null) ? 'w:val' : $attribute;
                $attributeValue = $xmlReader->getAttribute($attribute, $node);

                $styleValue = $this->readStyleDef($method, $attributeValue, $expected);
                if ($styleValue !== null) {
                    $styles[$styleProp] = $styleValue;
                }
            }
        }

        return $styles;
    }

    /**
     * Return style definition based on conversion method
     *
     * @param string $method
     * @ignoreScrutinizerPatch
     * @param string|null $attributeValue
     * @param mixed $expected
     * @return mixed
     */
    private function readStyleDef($method, $attributeValue, $expected)
    {
        $style = $attributeValue;

        if (self::READ_SIZE == $method) {
            $style = $attributeValue / 2;
        } elseif (self::READ_TRUE == $method) {
            $style = $this->isOn($attributeValue);
        } elseif (self::READ_FALSE == $method) {
            $style = !$this->isOn($attributeValue);
        } elseif (self::READ_EQUAL == $method) {
            $style = $attributeValue == $expected;
        }

        return $style;
    }

    /**
     * Parses the value of the on/off value, null is considered true as it means the w:val attribute was not present
     *
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_OnOff.html
     * @param string $value
     * @return bool
     */
    private function isOn($value = null)
    {
        return $value === null || $value === '1' || $value === 'true' || $value === 'on';
    }

    /**
     * Returns the target of image, object, or link as stored in ::readMainRels
     *
     * @param string $docPart
     * @param string $rId
     * @return string|null
     */
    private function getMediaTarget($docPart, $rId)
    {
        $target = null;

        if (isset($this->rels[$docPart]) && isset($this->rels[$docPart][$rId])) {
            $target = $this->rels[$docPart][$rId]['target'];
        }

        return $target;
    }

    /**
     * Returns the target mode
     *
     * @param string $docPart
     * @param string $rId
     * @return string|null
     */
    private function getTargetMode($docPart, $rId)
    {
        $mode = null;

        if (isset($this->rels[$docPart]) && isset($this->rels[$docPart][$rId])) {
            $mode = $this->rels[$docPart][$rId]['targetMode'];
        }

        return $mode;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/DocPropsApp.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

/**
 * Extended properties reader
 *
 * @since 0.10.0
 */
class DocPropsApp extends DocPropsCore
{
    /**
     * Property mapping
     *
     * @var array
     */
    protected $mapping = array('Company' => 'setCompany', 'Manager' => 'setManager');

    /**
     * Callback functions
     *
     * @var array
     */
    protected $callbacks = array();
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/DocPropsCore.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Core properties reader
 *
 * @since 0.10.0
 */
class DocPropsCore extends AbstractPart
{
    /**
     * Property mapping
     *
     * @var array
     */
    protected $mapping = array(
        'dc:creator'        => 'setCreator',
        'dc:title'          => 'setTitle',
        'dc:description'    => 'setDescription',
        'dc:subject'        => 'setSubject',
        'cp:keywords'       => 'setKeywords',
        'cp:category'       => 'setCategory',
        'cp:lastModifiedBy' => 'setLastModifiedBy',
        'dcterms:created'   => 'setCreated',
        'dcterms:modified'  => 'setModified',
    );

    /**
     * Callback functions
     *
     * @var array
     */
    protected $callbacks = array('dcterms:created' => 'strtotime', 'dcterms:modified' => 'strtotime');

    /**
     * Read core/extended document properties.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);

        $docProps = $phpWord->getDocInfo();

        $nodes = $xmlReader->getElements('*');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                if (!isset($this->mapping[$node->nodeName])) {
                    continue;
                }
                $method = $this->mapping[$node->nodeName];
                $value = $node->nodeValue == '' ? null : $node->nodeValue;
                if (isset($this->callbacks[$node->nodeName])) {
                    $value = $this->callbacks[$node->nodeName]($value);
                }
                if (method_exists($docProps, $method)) {
                    $docProps->$method($value);
                }
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/DocPropsCustom.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\Metadata\DocInfo;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Custom properties reader
 *
 * @since 0.11.0
 */
class DocPropsCustom extends AbstractPart
{
    /**
     * Read custom document properties.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);
        $docProps = $phpWord->getDocInfo();

        $nodes = $xmlReader->getElements('*');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $propertyName = $xmlReader->getAttribute('name', $node);
                $attributeNode = $xmlReader->getElement('*', $node);
                $attributeType = $attributeNode->nodeName;
                $attributeValue = $attributeNode->nodeValue;
                $attributeValue = DocInfo::convertProperty($attributeValue, $attributeType);
                $attributeType = DocInfo::convertPropertyType($attributeType);
                $docProps->setCustomProperty($propertyName, $attributeValue, $attributeType);
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Document.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\Element\Section;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Document reader
 *
 * @since 0.10.0
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod) For readWPNode
 */
class Document extends AbstractPart
{
    /**
     * PhpWord object
     *
     * @var \PhpOffice\PhpWord\PhpWord
     */
    private $phpWord;

    /**
     * Read document.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $this->phpWord = $phpWord;
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);
        $readMethods = array('w:p' => 'readWPNode', 'w:tbl' => 'readTable', 'w:sectPr' => 'readWSectPrNode');

        $nodes = $xmlReader->getElements('w:body/*');
        if ($nodes->length > 0) {
            $section = $this->phpWord->addSection();
            foreach ($nodes as $node) {
                if (isset($readMethods[$node->nodeName])) {
                    $readMethod = $readMethods[$node->nodeName];
                    $this->$readMethod($xmlReader, $node, $section);
                }
            }
        }
    }

    /**
     * Read header footer.
     *
     * @param array $settings
     * @param \PhpOffice\PhpWord\Element\Section &$section
     */
    private function readHeaderFooter($settings, Section &$section)
    {
        $readMethods = array('w:p' => 'readParagraph', 'w:tbl' => 'readTable');

        if (is_array($settings) && isset($settings['hf'])) {
            foreach ($settings['hf'] as $rId => $hfSetting) {
                if (isset($this->rels['document'][$rId])) {
                    list($hfType, $xmlFile, $docPart) = array_values($this->rels['document'][$rId]);
                    $addMethod = "add{$hfType}";
                    $hfObject = $section->$addMethod($hfSetting['type']);

                    // Read header/footer content
                    $xmlReader = new XMLReader();
                    $xmlReader->getDomFromZip($this->docFile, $xmlFile);
                    $nodes = $xmlReader->getElements('*');
                    if ($nodes->length > 0) {
                        foreach ($nodes as $node) {
                            if (isset($readMethods[$node->nodeName])) {
                                $readMethod = $readMethods[$node->nodeName];
                                $this->$readMethod($xmlReader, $node, $hfObject, $docPart);
                            }
                        }
                    }
                }
            }
        }
    }

    /**
     * Read w:sectPr
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $domNode
     * @ignoreScrutinizerPatch
     * @return array
     */
    private function readSectionStyle(XMLReader $xmlReader, \DOMElement $domNode)
    {
        $styleDefs = array(
            'breakType'     => array(self::READ_VALUE, 'w:type'),
            'vAlign'        => array(self::READ_VALUE, 'w:vAlign'),
            'pageSizeW'     => array(self::READ_VALUE, 'w:pgSz', 'w:w'),
            'pageSizeH'     => array(self::READ_VALUE, 'w:pgSz', 'w:h'),
            'orientation'   => array(self::READ_VALUE, 'w:pgSz', 'w:orient'),
            'colsNum'       => array(self::READ_VALUE, 'w:cols', 'w:num'),
            'colsSpace'     => array(self::READ_VALUE, 'w:cols', 'w:space'),
            'marginTop'     => array(self::READ_VALUE, 'w:pgMar', 'w:top'),
            'marginLeft'    => array(self::READ_VALUE, 'w:pgMar', 'w:left'),
            'marginBottom'  => array(self::READ_VALUE, 'w:pgMar', 'w:bottom'),
            'marginRight'   => array(self::READ_VALUE, 'w:pgMar', 'w:right'),
            'headerHeight'  => array(self::READ_VALUE, 'w:pgMar', 'w:header'),
            'footerHeight'  => array(self::READ_VALUE, 'w:pgMar', 'w:footer'),
            'gutter'        => array(self::READ_VALUE, 'w:pgMar', 'w:gutter'),
        );
        $styles = $this->readStyleDefs($xmlReader, $domNode, $styleDefs);

        // Header and footer
        // @todo Cleanup this part
        $nodes = $xmlReader->getElements('*', $domNode);
        foreach ($nodes as $node) {
            if ($node->nodeName == 'w:headerReference' || $node->nodeName == 'w:footerReference') {
                $id = $xmlReader->getAttribute('r:id', $node);
                $styles['hf'][$id] = array(
                    'method' => str_replace('w:', '', str_replace('Reference', '', $node->nodeName)),
                    'type'   => $xmlReader->getAttribute('w:type', $node),
                );
            }
        }

        return $styles;
    }

    /**
     * Read w:p node.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $node
     * @param \PhpOffice\PhpWord\Element\Section &$section
     *
     * @todo <w:lastRenderedPageBreak>
     */
    private function readWPNode(XMLReader $xmlReader, \DOMElement $node, Section &$section)
    {
        // Page break
        if ($xmlReader->getAttribute('w:type', $node, 'w:r/w:br') == 'page') {
            $section->addPageBreak(); // PageBreak
        }

        // Paragraph
        $this->readParagraph($xmlReader, $node, $section);

        // Section properties
        if ($xmlReader->elementExists('w:pPr/w:sectPr', $node)) {
            $sectPrNode = $xmlReader->getElement('w:pPr/w:sectPr', $node);
            if ($sectPrNode !== null) {
                $this->readWSectPrNode($xmlReader, $sectPrNode, $section);
            }
            $section = $this->phpWord->addSection();
        }
    }

    /**
     * Read w:sectPr node.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $node
     * @param \PhpOffice\PhpWord\Element\Section &$section
     */
    private function readWSectPrNode(XMLReader $xmlReader, \DOMElement $node, Section &$section)
    {
        $style = $this->readSectionStyle($xmlReader, $node);
        $section->setStyle($style);
        $this->readHeaderFooter($style, $section);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Endnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

/**
 * Endnotes reader
 *
 * @since 0.10.0
 */
class Endnotes extends Footnotes
{
    /**
     * Collection name
     *
     * @var string
     */
    protected $collection = 'endnotes';

    /**
     * Element name
     *
     * @var string
     */
    protected $element = 'endnote';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Footnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Footnotes reader
 *
 * @since 0.10.0
 */
class Footnotes extends AbstractPart
{
    /**
     * Collection name footnotes|endnotes
     *
     * @var string
     */
    protected $collection = 'footnotes';

    /**
     * Element name footnote|endnote
     *
     * @var string
     */
    protected $element = 'footnote';

    /**
     * Read (footnotes|endnotes).xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);
        $nodes = $xmlReader->getElements('*');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $id = $xmlReader->getAttribute('w:id', $node);
                $type = $xmlReader->getAttribute('w:type', $node);

                // Avoid w:type "separator" and "continuationSeparator"
                // Only look for <footnote> or <endnote> without w:type attribute, or with w:type = normal
                if ((is_null($type) || $type === 'normal')) {
                    $element = $this->getElement($phpWord, $id);
                    if ($element !== null) {
                        $pNodes = $xmlReader->getElements('w:p/*', $node);
                        foreach ($pNodes as $pNode) {
                            $this->readRun($xmlReader, $pNode, $element, $this->collection);
                        }
                        $addMethod = "add{$this->element}";
                        $phpWord->$addMethod($element);
                    }
                }
            }
        }
    }

    /**
     * Searches for the element with the given relationId
     *
     * @param PhpWord $phpWord
     * @param int $relationId
     * @return \PhpOffice\PhpWord\Element\AbstractContainer|null
     */
    private function getElement(PhpWord $phpWord, $relationId)
    {
        $getMethod = "get{$this->collection}";
        $collection = $phpWord->$getMethod()->getItems();

        //not found by key, looping to search by relationId
        foreach ($collection as $collectionElement) {
            if ($collectionElement->getRelationId() == $relationId) {
                return $collectionElement;
            }
        }

        return null;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Numbering.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;

/**
 * Numbering reader
 *
 * @since 0.10.0
 */
class Numbering extends AbstractPart
{
    /**
     * Read numbering.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $abstracts = array();
        $numberings = array();
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);

        // Abstract numbering definition
        $nodes = $xmlReader->getElements('w:abstractNum');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $abstractId = $xmlReader->getAttribute('w:abstractNumId', $node);
                $abstracts[$abstractId] = array('levels' => array());
                $abstract = &$abstracts[$abstractId];
                $subnodes = $xmlReader->getElements('*', $node);
                foreach ($subnodes as $subnode) {
                    switch ($subnode->nodeName) {
                        case 'w:multiLevelType':
                            $abstract['type'] = $xmlReader->getAttribute('w:val', $subnode);
                            break;
                        case 'w:lvl':
                            $levelId = $xmlReader->getAttribute('w:ilvl', $subnode);
                            $abstract['levels'][$levelId] = $this->readLevel($xmlReader, $subnode, $levelId);
                            break;
                    }
                }
            }
        }

        // Numbering instance definition
        $nodes = $xmlReader->getElements('w:num');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $numId = $xmlReader->getAttribute('w:numId', $node);
                $abstractId = $xmlReader->getAttribute('w:val', $node, 'w:abstractNumId');
                $numberings[$numId] = $abstracts[$abstractId];
                $numberings[$numId]['numId'] = $numId;
                $subnodes = $xmlReader->getElements('w:lvlOverride/w:lvl', $node);
                foreach ($subnodes as $subnode) {
                    $levelId = $xmlReader->getAttribute('w:ilvl', $subnode);
                    $overrides = $this->readLevel($xmlReader, $subnode, $levelId);
                    foreach ($overrides as $key => $value) {
                        $numberings[$numId]['levels'][$levelId][$key] = $value;
                    }
                }
            }
        }

        // Push to Style collection
        foreach ($numberings as $numId => $numbering) {
            $phpWord->addNumberingStyle("PHPWordList{$numId}", $numbering);
        }
    }

    /**
     * Read numbering level definition from w:abstractNum and w:num
     *
     * @param \PhpOffice\PhpWord\Shared\XMLReader $xmlReader
     * @param \DOMElement $subnode
     * @param int $levelId
     * @return array
     */
    private function readLevel(XMLReader $xmlReader, \DOMElement $subnode, $levelId)
    {
        $level = array();

        $level['level'] = $levelId;
        $level['start'] = $xmlReader->getAttribute('w:val', $subnode, 'w:start');
        $level['format'] = $xmlReader->getAttribute('w:val', $subnode, 'w:numFmt');
        $level['restart'] = $xmlReader->getAttribute('w:val', $subnode, 'w:lvlRestart');
        $level['suffix'] = $xmlReader->getAttribute('w:val', $subnode, 'w:suff');
        $level['text'] = $xmlReader->getAttribute('w:val', $subnode, 'w:lvlText');
        $level['alignment'] = $xmlReader->getAttribute('w:val', $subnode, 'w:lvlJc');
        $level['tab'] = $xmlReader->getAttribute('w:pos', $subnode, 'w:pPr/w:tabs/w:tab');
        $level['left'] = $xmlReader->getAttribute('w:left', $subnode, 'w:pPr/w:ind');
        $level['hanging'] = $xmlReader->getAttribute('w:hanging', $subnode, 'w:pPr/w:ind');
        $level['font'] = $xmlReader->getAttribute('w:ascii', $subnode, 'w:rPr/w:rFonts');
        $level['hint'] = $xmlReader->getAttribute('w:hint', $subnode, 'w:rPr/w:rFonts');

        foreach ($level as $key => $value) {
            if (is_null($value)) {
                unset($level[$key]);
            }
        }

        return $level;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Settings.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\ComplexType\TrackChangesView;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;
use PhpOffice\PhpWord\Style\Language;

/**
 * Settings reader
 *
 * @since 0.14.0
 */
class Settings extends AbstractPart
{
    private static $booleanProperties = array(
        'mirrorMargins',
        'hideSpellingErrors',
        'hideGrammaticalErrors',
        'trackRevisions',
        'doNotTrackMoves',
        'doNotTrackFormatting',
        'evenAndOddHeaders',
        'updateFields',
        'autoHyphenation',
        'doNotHyphenateCaps',
    );

    /**
     * Read settings.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);

        $docSettings = $phpWord->getSettings();

        $nodes = $xmlReader->getElements('*');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $name = str_replace('w:', '', $node->nodeName);
                $value = $xmlReader->getAttribute('w:val', $node);
                $method = 'set' . $name;

                if (in_array($name, $this::$booleanProperties)) {
                    if ($value == 'false') {
                        $docSettings->$method(false);
                    } else {
                        $docSettings->$method(true);
                    }
                } elseif (method_exists($this, $method)) {
                    $this->$method($xmlReader, $phpWord, $node);
                } elseif (method_exists($docSettings, $method)) {
                    $docSettings->$method($value);
                }
            }
        }
    }

    /**
     * Sets the document Language
     *
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setThemeFontLang(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $val = $xmlReader->getAttribute('w:val', $node);
        $eastAsia = $xmlReader->getAttribute('w:eastAsia', $node);
        $bidi = $xmlReader->getAttribute('w:bidi', $node);

        $themeFontLang = new Language();
        $themeFontLang->setLatin($val);
        $themeFontLang->setEastAsia($eastAsia);
        $themeFontLang->setBidirectional($bidi);

        $phpWord->getSettings()->setThemeFontLang($themeFontLang);
    }

    /**
     * Sets the document protection
     *
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setDocumentProtection(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $documentProtection = $phpWord->getSettings()->getDocumentProtection();

        $edit = $xmlReader->getAttribute('w:edit', $node);
        if ($edit !== null) {
            $documentProtection->setEditing($edit);
        }
    }

    /**
     * Sets the proof state
     *
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setProofState(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $proofState = $phpWord->getSettings()->getProofState();

        $spelling = $xmlReader->getAttribute('w:spelling', $node);
        $grammar = $xmlReader->getAttribute('w:grammar', $node);

        if ($spelling !== null) {
            $proofState->setSpelling($spelling);
        }
        if ($grammar !== null) {
            $proofState->setGrammar($grammar);
        }
    }

    /**
     * Sets the proof state
     *
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setZoom(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $percent = $xmlReader->getAttribute('w:percent', $node);
        $val = $xmlReader->getAttribute('w:val', $node);

        if ($percent !== null || $val !== null) {
            $phpWord->getSettings()->setZoom($percent === null ? $val : $percent);
        }
    }

    /**
     * Set the Revision view
     *
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setRevisionView(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $revisionView = new TrackChangesView();
        $revisionView->setMarkup(filter_var($xmlReader->getAttribute('w:markup', $node), FILTER_VALIDATE_BOOLEAN));
        $revisionView->setComments($xmlReader->getAttribute('w:comments', $node));
        $revisionView->setInsDel(filter_var($xmlReader->getAttribute('w:insDel', $node), FILTER_VALIDATE_BOOLEAN));
        $revisionView->setFormatting(filter_var($xmlReader->getAttribute('w:formatting', $node), FILTER_VALIDATE_BOOLEAN));
        $revisionView->setInkAnnotations(filter_var($xmlReader->getAttribute('w:inkAnnotations', $node), FILTER_VALIDATE_BOOLEAN));
        $phpWord->getSettings()->setRevisionView($revisionView);
    }

    /**
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setConsecutiveHyphenLimit(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $value = $xmlReader->getAttribute('w:val', $node);

        if ($value !== null) {
            $phpWord->getSettings()->setConsecutiveHyphenLimit($value);
        }
    }

    /**
     * @param XMLReader $xmlReader
     * @param PhpWord $phpWord
     * @param \DOMElement $node
     */
    protected function setHyphenationZone(XMLReader $xmlReader, PhpWord $phpWord, \DOMElement $node)
    {
        $value = $xmlReader->getAttribute('w:val', $node);

        if ($value !== null) {
            $phpWord->getSettings()->setHyphenationZone($value);
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007/Styles.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader\Word2007;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;
use PhpOffice\PhpWord\Style\Language;

/**
 * Styles reader
 *
 * @since 0.10.0
 */
class Styles extends AbstractPart
{
    /**
     * Read styles.xml.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function read(PhpWord $phpWord)
    {
        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($this->docFile, $this->xmlFile);

        $fontDefaults = $xmlReader->getElement('w:docDefaults/w:rPrDefault');
        if ($fontDefaults !== null) {
            $fontDefaultStyle = $this->readFontStyle($xmlReader, $fontDefaults);
            if (array_key_exists('name', $fontDefaultStyle)) {
                $phpWord->setDefaultFontName($fontDefaultStyle['name']);
            }
            if (array_key_exists('size', $fontDefaultStyle)) {
                $phpWord->setDefaultFontSize($fontDefaultStyle['size']);
            }
            if (array_key_exists('lang', $fontDefaultStyle)) {
                $phpWord->getSettings()->setThemeFontLang(new Language($fontDefaultStyle['lang']));
            }
        }

        $paragraphDefaults = $xmlReader->getElement('w:docDefaults/w:pPrDefault');
        if ($paragraphDefaults !== null) {
            $paragraphDefaultStyle = $this->readParagraphStyle($xmlReader, $paragraphDefaults);
            if ($paragraphDefaultStyle != null) {
                $phpWord->setDefaultParagraphStyle($paragraphDefaultStyle);
            }
        }

        $nodes = $xmlReader->getElements('w:style');
        if ($nodes->length > 0) {
            foreach ($nodes as $node) {
                $type = $xmlReader->getAttribute('w:type', $node);
                $name = $xmlReader->getAttribute('w:val', $node, 'w:name');
                if (is_null($name)) {
                    $name = $xmlReader->getAttribute('w:styleId', $node);
                }
                $headingMatches = array();
                preg_match('/Heading\s*(\d)/i', $name, $headingMatches);
                // $default = ($xmlReader->getAttribute('w:default', $node) == 1);
                switch ($type) {
                    case 'paragraph':
                        $paragraphStyle = $this->readParagraphStyle($xmlReader, $node);
                        $fontStyle = $this->readFontStyle($xmlReader, $node);
                        if (!empty($headingMatches)) {
                            $phpWord->addTitleStyle($headingMatches[1], $fontStyle, $paragraphStyle);
                        } else {
                            if (empty($fontStyle)) {
                                if (is_array($paragraphStyle)) {
                                    $phpWord->addParagraphStyle($name, $paragraphStyle);
                                }
                            } else {
                                $phpWord->addFontStyle($name, $fontStyle, $paragraphStyle);
                            }
                        }
                        break;
                    case 'character':
                        $fontStyle = $this->readFontStyle($xmlReader, $node);
                        if (!empty($fontStyle)) {
                            $phpWord->addFontStyle($name, $fontStyle);
                        }
                        break;
                    case 'table':
                        $tStyle = $this->readTableStyle($xmlReader, $node);
                        if (!empty($tStyle)) {
                            $phpWord->addTableStyle($name, $tStyle);
                        }
                        break;
                }
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Reader/Word2007.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Reader;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLReader;
use PhpOffice\PhpWord\Shared\ZipArchive;

/**
 * Reader for Word2007
 *
 * @since 0.8.0
 * @todo watermark, checkbox, toc
 * @todo Partly done: image, object
 */
class Word2007 extends AbstractReader implements ReaderInterface
{
    /**
     * Loads PhpWord from file
     *
     * @param string $docFile
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function load($docFile)
    {
        $phpWord = new PhpWord();
        $relationships = $this->readRelationships($docFile);

        $steps = array(
            array('stepPart' => 'document', 'stepItems' => array(
                'styles'    => 'Styles',
                'numbering' => 'Numbering',
            )),
            array('stepPart' => 'main', 'stepItems' => array(
                'officeDocument'      => 'Document',
                'core-properties'     => 'DocPropsCore',
                'extended-properties' => 'DocPropsApp',
                'custom-properties'   => 'DocPropsCustom',
            )),
            array('stepPart' => 'document', 'stepItems' => array(
                'endnotes'  => 'Endnotes',
                'footnotes' => 'Footnotes',
                'settings'  => 'Settings',
            )),
        );

        foreach ($steps as $step) {
            $stepPart = $step['stepPart'];
            $stepItems = $step['stepItems'];
            if (!isset($relationships[$stepPart])) {
                continue;
            }
            foreach ($relationships[$stepPart] as $relItem) {
                $relType = $relItem['type'];
                if (isset($stepItems[$relType])) {
                    $partName = $stepItems[$relType];
                    $xmlFile = $relItem['target'];
                    $this->readPart($phpWord, $relationships, $partName, $docFile, $xmlFile);
                }
            }
        }

        return $phpWord;
    }

    /**
     * Read document part.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     * @param array $relationships
     * @param string $partName
     * @param string $docFile
     * @param string $xmlFile
     */
    private function readPart(PhpWord $phpWord, $relationships, $partName, $docFile, $xmlFile)
    {
        $partClass = "PhpOffice\\PhpWord\\Reader\\Word2007\\{$partName}";
        if (class_exists($partClass)) {
            /** @var \PhpOffice\PhpWord\Reader\Word2007\AbstractPart $part Type hint */
            $part = new $partClass($docFile, $xmlFile);
            $part->setRels($relationships);
            $part->read($phpWord);
        }
    }

    /**
     * Read all relationship files
     *
     * @param string $docFile
     * @return array
     */
    private function readRelationships($docFile)
    {
        $relationships = array();

        // _rels/.rels
        $relationships['main'] = $this->getRels($docFile, '_rels/.rels');

        // word/_rels/*.xml.rels
        $wordRelsPath = 'word/_rels/';
        $zip = new ZipArchive();
        if ($zip->open($docFile) === true) {
            for ($i = 0; $i < $zip->numFiles; $i++) {
                $xmlFile = $zip->getNameIndex($i);
                if ((substr($xmlFile, 0, strlen($wordRelsPath))) == $wordRelsPath && (substr($xmlFile, -1)) != '/') {
                    $docPart = str_replace('.xml.rels', '', str_replace($wordRelsPath, '', $xmlFile));
                    $relationships[$docPart] = $this->getRels($docFile, $xmlFile, 'word/');
                }
            }
            $zip->close();
        }

        return $relationships;
    }

    /**
     * Get relationship array
     *
     * @param string $docFile
     * @param string $xmlFile
     * @param string $targetPrefix
     * @return array
     */
    private function getRels($docFile, $xmlFile, $targetPrefix = '')
    {
        $metaPrefix = 'http://schemas.openxmlformats.org/package/2006/relationships/metadata/';
        $officePrefix = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships/';

        $rels = array();

        $xmlReader = new XMLReader();
        $xmlReader->getDomFromZip($docFile, $xmlFile);
        $nodes = $xmlReader->getElements('*');
        foreach ($nodes as $node) {
            $rId = $xmlReader->getAttribute('Id', $node);
            $type = $xmlReader->getAttribute('Type', $node);
            $target = $xmlReader->getAttribute('Target', $node);
            $mode = $xmlReader->getAttribute('TargetMode', $node);

            // Remove URL prefixes from $type to make it easier to read
            $type = str_replace($metaPrefix, '', $type);
            $type = str_replace($officePrefix, '', $type);
            $docPart = str_replace('.xml', '', $target);

            // Do not add prefix to link source
            if ($type != 'hyperlink' && $mode != 'External') {
                $target = $targetPrefix . $target;
            }

            // Push to return array
            $rels[$rId] = array('type' => $type, 'target' => $target, 'docPart' => $docPart, 'targetMode' => $mode);
        }
        ksort($rels);

        return $rels;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/resources/doc.png

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/resources/ppt.png

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/resources/xls.png

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Settings.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

/**
 * PHPWord settings class
 *
 * @since 0.8.0
 */
class Settings
{
    /**
     * Zip libraries
     *
     * @const string
     */
    const ZIPARCHIVE = 'ZipArchive';
    const PCLZIP = 'PclZip';
    const OLD_LIB = 'PhpOffice\\PhpWord\\Shared\\ZipArchive'; // @deprecated 0.11

    /**
     * PDF rendering libraries
     *
     * @const string
     */
    const PDF_RENDERER_DOMPDF = 'DomPDF';
    const PDF_RENDERER_TCPDF = 'TCPDF';
    const PDF_RENDERER_MPDF = 'MPDF';

    /**
     * Measurement units multiplication factor
     *
     * Applied to:
     * - Section: margins, header/footer height, gutter, column spacing
     * - Tab: position
     * - Indentation: left, right, firstLine, hanging
     * - Spacing: before, after
     *
     * @const string
     */
    const UNIT_TWIP = 'twip'; // = 1/20 point
    const UNIT_CM = 'cm';
    const UNIT_MM = 'mm';
    const UNIT_INCH = 'inch';
    const UNIT_POINT = 'point'; // = 1/72 inch
    const UNIT_PICA = 'pica'; // = 1/6 inch = 12 points

    /**
     * Default font settings
     *
     * OOXML defined font size values in halfpoints, i.e. twice of what PhpWord
     * use, and the conversion will be conducted during XML writing.
     */
    const DEFAULT_FONT_NAME = 'Arial';
    const DEFAULT_FONT_SIZE = 10;
    const DEFAULT_FONT_COLOR = '000000';
    const DEFAULT_FONT_CONTENT_TYPE = 'default'; // default|eastAsia|cs
    const DEFAULT_PAPER = 'A4';

    /**
     * Compatibility option for XMLWriter
     *
     * @var bool
     */
    private static $xmlWriterCompatibility = true;

    /**
     * Name of the class used for Zip file management
     *
     * @var string
     */
    private static $zipClass = self::ZIPARCHIVE;

    /**
     * Name of the external Library used for rendering PDF files
     *
     * @var string
     */
    private static $pdfRendererName = null;

    /**
     * Directory Path to the external Library used for rendering PDF files
     *
     * @var string
     */
    private static $pdfRendererPath = null;

    /**
     * Measurement unit
     *
     * @var int|float
     */
    private static $measurementUnit = self::UNIT_TWIP;

    /**
     * Default font name
     *
     * @var string
     */
    private static $defaultFontName = self::DEFAULT_FONT_NAME;

    /**
     * Default font size
     * @var int
     */
    private static $defaultFontSize = self::DEFAULT_FONT_SIZE;

    /**
     * Default paper
     * @var string
     */
    private static $defaultPaper = self::DEFAULT_PAPER;

    /**
     * The user defined temporary directory.
     *
     * @var string
     */
    private static $tempDir = '';

    /**
     * Enables built-in output escaping mechanism.
     * Default value is `false` for backward compatibility with versions below 0.13.0.
     *
     * @var bool
     */
    private static $outputEscapingEnabled = false;

    /**
     * Return the compatibility option used by the XMLWriter
     *
     * @return bool Compatibility
     */
    public static function hasCompatibility()
    {
        return self::$xmlWriterCompatibility;
    }

    /**
     * Set the compatibility option used by the XMLWriter
     *
     * This sets the setIndent and setIndentString for better compatibility
     *
     * @param bool $compatibility
     * @return bool
     */
    public static function setCompatibility($compatibility)
    {
        $compatibility = (bool) $compatibility;
        self::$xmlWriterCompatibility = $compatibility;

        return true;
    }

    /**
     * Get zip handler class
     *
     * @return string
     */
    public static function getZipClass()
    {
        return self::$zipClass;
    }

    /**
     * Set zip handler class
     *
     * @param  string $zipClass
     * @return bool
     */
    public static function setZipClass($zipClass)
    {
        if (in_array($zipClass, array(self::PCLZIP, self::ZIPARCHIVE, self::OLD_LIB))) {
            self::$zipClass = $zipClass;

            return true;
        }

        return false;
    }

    /**
     * Set details of the external library for rendering PDF files
     *
     * @param string $libraryName
     * @param string $libraryBaseDir
     * @return bool Success or failure
     */
    public static function setPdfRenderer($libraryName, $libraryBaseDir)
    {
        if (!self::setPdfRendererName($libraryName)) {
            return false;
        }

        return self::setPdfRendererPath($libraryBaseDir);
    }

    /**
     * Return the PDF Rendering Library.
     *
     * @return string
     */
    public static function getPdfRendererName()
    {
        return self::$pdfRendererName;
    }

    /**
     * Identify the external library to use for rendering PDF files
     *
     * @param string $libraryName
     * @return bool
     */
    public static function setPdfRendererName($libraryName)
    {
        $pdfRenderers = array(self::PDF_RENDERER_DOMPDF, self::PDF_RENDERER_TCPDF, self::PDF_RENDERER_MPDF);
        if (!in_array($libraryName, $pdfRenderers)) {
            return false;
        }
        self::$pdfRendererName = $libraryName;

        return true;
    }

    /**
     * Return the directory path to the PDF Rendering Library.
     *
     * @return string
     */
    public static function getPdfRendererPath()
    {
        return self::$pdfRendererPath;
    }

    /**
     * Location of external library to use for rendering PDF files
     *
     * @param string $libraryBaseDir Directory path to the library's base folder
     * @return bool Success or failure
     */
    public static function setPdfRendererPath($libraryBaseDir)
    {
        if (false === file_exists($libraryBaseDir) || false === is_readable($libraryBaseDir)) {
            return false;
        }
        self::$pdfRendererPath = $libraryBaseDir;

        return true;
    }

    /**
     * Get measurement unit
     *
     * @return string
     */
    public static function getMeasurementUnit()
    {
        return self::$measurementUnit;
    }

    /**
     * Set measurement unit
     *
     * @param string $value
     * @return bool
     */
    public static function setMeasurementUnit($value)
    {
        $units = array(self::UNIT_TWIP, self::UNIT_CM, self::UNIT_MM, self::UNIT_INCH,
            self::UNIT_POINT, self::UNIT_PICA, );
        if (!in_array($value, $units)) {
            return false;
        }
        self::$measurementUnit = $value;

        return true;
    }

    /**
     * Sets the user defined path to temporary directory.
     *
     * @since 0.12.0
     *
     * @param string $tempDir The user defined path to temporary directory
     */
    public static function setTempDir($tempDir)
    {
        self::$tempDir = $tempDir;
    }

    /**
     * Returns path to temporary directory.
     *
     * @since 0.12.0
     *
     * @return string
     */
    public static function getTempDir()
    {
        if (!empty(self::$tempDir)) {
            $tempDir = self::$tempDir;
        } else {
            $tempDir = sys_get_temp_dir();
        }

        return $tempDir;
    }

    /**
     * @since 0.13.0
     *
     * @return bool
     */
    public static function isOutputEscapingEnabled()
    {
        return self::$outputEscapingEnabled;
    }

    /**
     * @since 0.13.0
     *
     * @param bool $outputEscapingEnabled
     */
    public static function setOutputEscapingEnabled($outputEscapingEnabled)
    {
        self::$outputEscapingEnabled = $outputEscapingEnabled;
    }

    /**
     * Get default font name
     *
     * @return string
     */
    public static function getDefaultFontName()
    {
        return self::$defaultFontName;
    }

    /**
     * Set default font name
     *
     * @param string $value
     * @return bool
     */
    public static function setDefaultFontName($value)
    {
        if (is_string($value) && trim($value) !== '') {
            self::$defaultFontName = $value;

            return true;
        }

        return false;
    }

    /**
     * Get default font size
     *
     * @return int
     */
    public static function getDefaultFontSize()
    {
        return self::$defaultFontSize;
    }

    /**
     * Set default font size
     *
     * @param int $value
     * @return bool
     */
    public static function setDefaultFontSize($value)
    {
        $value = (int) $value;
        if ($value > 0) {
            self::$defaultFontSize = $value;

            return true;
        }

        return false;
    }

    /**
     * Load setting from phpword.yml or phpword.yml.dist
     *
     * @param string $filename
     * @return array
     */
    public static function loadConfig($filename = null)
    {
        // Get config file
        $configFile = null;
        $configPath = __DIR__ . '/../../';
        if ($filename !== null) {
            $files = array($filename);
        } else {
            $files = array("{$configPath}phpword.ini", "{$configPath}phpword.ini.dist");
        }
        foreach ($files as $file) {
            if (file_exists($file)) {
                $configFile = realpath($file);
                break;
            }
        }

        // Parse config file
        $config = array();
        if ($configFile !== null) {
            $config = @parse_ini_file($configFile);
            if ($config === false) {
                return $config;
            }
        }

        // Set config value
        foreach ($config as $key => $value) {
            $method = "set{$key}";
            if (method_exists(__CLASS__, $method)) {
                self::$method($value);
            }
        }

        return $config;
    }

    /**
     * Get default paper
     *
     * @return string
     */
    public static function getDefaultPaper()
    {
        return self::$defaultPaper;
    }

    /**
     * Set default paper
     *
     * @param string $value
     * @return bool
     */
    public static function setDefaultPaper($value)
    {
        if (is_string($value) && trim($value) !== '') {
            self::$defaultPaper = $value;

            return true;
        }

        return false;
    }

    /**
     * Return the compatibility option used by the XMLWriter
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public static function getCompatibility()
    {
        return self::hasCompatibility();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/AbstractEnum.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

abstract class AbstractEnum
{
    private static $constCacheArray = null;

    private static function getConstants()
    {
        if (self::$constCacheArray == null) {
            self::$constCacheArray = array();
        }
        $calledClass = get_called_class();
        if (!array_key_exists($calledClass, self::$constCacheArray)) {
            $reflect = new \ReflectionClass($calledClass);
            self::$constCacheArray[$calledClass] = $reflect->getConstants();
        }

        return self::$constCacheArray[$calledClass];
    }

    /**
     * Returns all values for this enum
     *
     * @return array
     */
    public static function values()
    {
        return array_values(self::getConstants());
    }

    /**
     * Returns true the value is valid for this enum
     *
     * @param string $value
     * @return bool true if value is valid
     */
    public static function isValid($value)
    {
        $values = array_values(self::getConstants());

        return in_array($value, $values, true);
    }

    /**
     * Validates that the value passed is a valid value
     *
     * @param string $value
     * @throws \InvalidArgumentException if the value passed is not valid for this enum
     */
    public static function validate($value)
    {
        if (!self::isValid($value)) {
            $calledClass = get_called_class();
            $values = array_values(self::getConstants());
            throw new \InvalidArgumentException("$value is not a valid value for $calledClass, possible values are " . implode(', ', $values));
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/Converter.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

/**
 * Common converter functions
 */
class Converter
{
    const INCH_TO_CM = 2.54;
    const INCH_TO_TWIP = 1440;
    const INCH_TO_PIXEL = 96;
    const INCH_TO_POINT = 72;
    const INCH_TO_PICA = 6;
    const PIXEL_TO_EMU = 9525;
    const DEGREE_TO_ANGLE = 60000;

    /**
     * Convert centimeter to twip
     *
     * @param float $centimeter
     * @return float
     */
    public static function cmToTwip($centimeter = 1)
    {
        return $centimeter / self::INCH_TO_CM * self::INCH_TO_TWIP;
    }

    /**
     * Convert centimeter to inch
     *
     * @param float $centimeter
     * @return float
     */
    public static function cmToInch($centimeter = 1)
    {
        return $centimeter / self::INCH_TO_CM;
    }

    /**
     * Convert centimeter to pixel
     *
     * @param float $centimeter
     * @return float
     */
    public static function cmToPixel($centimeter = 1)
    {
        return $centimeter / self::INCH_TO_CM * self::INCH_TO_PIXEL;
    }

    /**
     * Convert centimeter to point
     *
     * @param float $centimeter
     * @return float
     */
    public static function cmToPoint($centimeter = 1)
    {
        return $centimeter / self::INCH_TO_CM * self::INCH_TO_POINT;
    }

    /**
     * Convert centimeter to EMU
     *
     * @param float $centimeter
     * @return float
     */
    public static function cmToEmu($centimeter = 1)
    {
        return round($centimeter / self::INCH_TO_CM * self::INCH_TO_PIXEL * self::PIXEL_TO_EMU);
    }

    /**
     * Convert inch to twip
     *
     * @param float $inch
     * @return float
     */
    public static function inchToTwip($inch = 1)
    {
        return $inch * self::INCH_TO_TWIP;
    }

    /**
     * Convert inch to centimeter
     *
     * @param float $inch
     * @return float
     */
    public static function inchToCm($inch = 1)
    {
        return $inch * self::INCH_TO_CM;
    }

    /**
     * Convert inch to pixel
     *
     * @param float $inch
     * @return float
     */
    public static function inchToPixel($inch = 1)
    {
        return $inch * self::INCH_TO_PIXEL;
    }

    /**
     * Convert inch to point
     *
     * @param float $inch
     * @return float
     */
    public static function inchToPoint($inch = 1)
    {
        return $inch * self::INCH_TO_POINT;
    }

    /**
     * Convert inch to EMU
     *
     * @param float $inch
     * @return int
     */
    public static function inchToEmu($inch = 1)
    {
        return round($inch * self::INCH_TO_PIXEL * self::PIXEL_TO_EMU);
    }

    /**
     * Convert pixel to twip
     *
     * @param float $pixel
     * @return float
     */
    public static function pixelToTwip($pixel = 1)
    {
        return $pixel / self::INCH_TO_PIXEL * self::INCH_TO_TWIP;
    }

    /**
     * Convert pixel to centimeter
     *
     * @param float $pixel
     * @return float
     */
    public static function pixelToCm($pixel = 1)
    {
        return $pixel / self::INCH_TO_PIXEL * self::INCH_TO_CM;
    }

    /**
     * Convert pixel to point
     *
     * @param float $pixel
     * @return float
     */
    public static function pixelToPoint($pixel = 1)
    {
        return $pixel / self::INCH_TO_PIXEL * self::INCH_TO_POINT;
    }

    /**
     * Convert pixel to EMU
     *
     * @param float $pixel
     * @return int
     */
    public static function pixelToEmu($pixel = 1)
    {
        return round($pixel * self::PIXEL_TO_EMU);
    }

    /**
     * Convert point to twip unit
     *
     * @param float $point
     * @return float
     */
    public static function pointToTwip($point = 1)
    {
        return $point / self::INCH_TO_POINT * self::INCH_TO_TWIP;
    }

    /**
     * Convert point to pixel
     *
     * @param float $point
     * @return float
     */
    public static function pointToPixel($point = 1)
    {
        return $point / self::INCH_TO_POINT * self::INCH_TO_PIXEL;
    }

    /**
     * Convert point to EMU
     *
     * @param float $point
     * @return float
     */
    public static function pointToEmu($point = 1)
    {
        return round($point / self::INCH_TO_POINT * self::INCH_TO_PIXEL * self::PIXEL_TO_EMU);
    }

    /**
     * Convert point to cm
     *
     * @param float $point
     * @return float
     */
    public static function pointToCm($point = 1)
    {
        return $point / self::INCH_TO_POINT * self::INCH_TO_CM;
    }

    /**
     * Convert EMU to pixel
     *
     * @param float $emu
     * @return float
     */
    public static function emuToPixel($emu = 1)
    {
        return round($emu / self::PIXEL_TO_EMU);
    }

    /**
     * Convert pica to point
     *
     * @param float $pica
     * @return float
     */
    public static function picaToPoint($pica = 1)
    {
        return $pica / self::INCH_TO_PICA * self::INCH_TO_POINT;
    }

    /**
     * Convert degree to angle
     *
     * @param float $degree
     * @return int
     */
    public static function degreeToAngle($degree = 1)
    {
        return (int) round($degree * self::DEGREE_TO_ANGLE);
    }

    /**
     * Convert angle to degrees
     *
     * @param float $angle
     * @return int
     */
    public static function angleToDegree($angle = 1)
    {
        return round($angle / self::DEGREE_TO_ANGLE);
    }

    /**
     * Convert colorname as string to RGB
     *
     * @param string $value color name
     * @return string color as hex RGB string, or original value if unknown
     */
    public static function stringToRgb($value)
    {
        switch ($value) {
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_YELLOW:
                return 'FFFF00';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_LIGHTGREEN:
                return '90EE90';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_CYAN:
                return '00FFFF';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_MAGENTA:
                return 'FF00FF';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_BLUE:
                return '0000FF';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_RED:
                return 'FF0000';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKBLUE:
                return '00008B';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKCYAN:
                return '008B8B';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKGREEN:
                return '006400';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKMAGENTA:
                return '8B008B';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKRED:
                return '8B0000';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKYELLOW:
                return '8B8B00';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_DARKGRAY:
                return 'A9A9A9';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_LIGHTGRAY:
                return 'D3D3D3';
            case \PhpOffice\PhpWord\Style\Font::FGCOLOR_BLACK:
                return '000000';
        }

        return $value;
    }

    /**
     * Convert HTML hexadecimal to RGB
     *
     * @param string $value HTML Color in hexadecimal
     * @return array Value in RGB
     */
    public static function htmlToRgb($value)
    {
        if ($value[0] == '#') {
            $value = substr($value, 1);
        } else {
            $value = self::stringToRgb($value);
        }

        if (strlen($value) == 6) {
            list($red, $green, $blue) = array($value[0] . $value[1], $value[2] . $value[3], $value[4] . $value[5]);
        } elseif (strlen($value) == 3) {
            list($red, $green, $blue) = array($value[0] . $value[0], $value[1] . $value[1], $value[2] . $value[2]);
        } else {
            return false;
        }

        $red = ctype_xdigit($red) ? hexdec($red) : 0;
        $green = ctype_xdigit($green) ? hexdec($green) : 0;
        $blue = ctype_xdigit($blue) ? hexdec($blue) : 0;

        return array($red, $green, $blue);
    }

    /**
     * Transforms a size in CSS format (eg. 10px, 10px, ...) to points
     *
     * @param string $value
     * @return float
     */
    public static function cssToPoint($value)
    {
        if ($value == '0') {
            return 0;
        }
        $matches = array();
        if (preg_match('/^[+-]?([0-9]+\.?[0-9]*)?(px|em|ex|%|in|cm|mm|pt|pc)$/i', $value, $matches)) {
            $size = $matches[1];
            $unit = $matches[2];

            switch ($unit) {
                case 'pt':
                    return $size;
                case 'px':
                    return self::pixelToPoint($size);
                case 'cm':
                    return self::cmToPoint($size);
                case 'mm':
                    return self::cmToPoint($size / 10);
                case 'in':
                    return self::inchToPoint($size);
                case 'pc':
                    return self::picaToPoint($size);
            }
        }

        return null;
    }

    /**
     * Transforms a size in CSS format (eg. 10px, 10px, ...) to twips
     *
     * @param string $value
     * @return float
     */
    public static function cssToTwip($value)
    {
        return self::pointToTwip(self::cssToPoint($value));
    }

    /**
     * Transforms a size in CSS format (eg. 10px, 10px, ...) to pixel
     *
     * @param string $value
     * @return float
     */
    public static function cssToPixel($value)
    {
        return self::pointToPixel(self::cssToPoint($value));
    }

    /**
     * Transforms a size in CSS format (eg. 10px, 10px, ...) to cm
     *
     * @param string $value
     * @return float
     */
    public static function cssToCm($value)
    {
        return self::pointToCm(self::cssToPoint($value));
    }

    /**
     * Transforms a size in CSS format (eg. 10px, 10px, ...) to emu
     *
     * @param string $value
     * @return float
     */
    public static function cssToEmu($value)
    {
        return self::pointToEmu(self::cssToPoint($value));
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/Drawing.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

/**
 * Drawing
 */
class Drawing
{
    const DPI_96 = 96;

    /**
     * Convert pixels to EMU
     *
     * @param  int $pValue Value in pixels
     * @return int
     */
    public static function pixelsToEmu($pValue = 0)
    {
        return round($pValue * 9525);
    }

    /**
     * Convert EMU to pixels
     *
     * @param  int $pValue Value in EMU
     * @return int
     */
    public static function emuToPixels($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return round($pValue / 9525);
    }

    /**
     * Convert pixels to points
     *
     * @param  int $pValue Value in pixels
     * @return float
     */
    public static function pixelsToPoints($pValue = 0)
    {
        return $pValue * 0.67777777;
    }

    /**
     * Convert points width to centimeters
     *
     * @param  int $pValue Value in points
     * @return float
     */
    public static function pointsToCentimeters($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return (($pValue * 1.333333333) / self::DPI_96) * 2.54;
    }

    /**
     * Convert points width to pixels
     *
     * @param  int $pValue Value in points
     * @return float
     */
    public static function pointsToPixels($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return $pValue * 1.333333333;
    }

    /**
     * Convert pixels to centimeters
     *
     * @param  int $pValue Value in pixels
     * @return float
     */
    public static function pixelsToCentimeters($pValue = 0)
    {
        //return $pValue * 0.028;
        return ($pValue / self::DPI_96) * 2.54;
    }

    /**
     * Convert centimeters width to pixels
     *
     * @param  int $pValue Value in centimeters
     * @return float
     */
    public static function centimetersToPixels($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return ($pValue / 2.54) * self::DPI_96;
    }

    /**
     * Convert degrees to angle
     *
     * @param  int $pValue Degrees
     * @return int
     */
    public static function degreesToAngle($pValue = 0)
    {
        return (int) round($pValue * 60000);
    }

    /**
     * Convert angle to degrees
     *
     * @param  int $pValue Angle
     * @return int
     */
    public static function angleToDegrees($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return round($pValue / 60000);
    }

    /**
     * Convert centimeters width to twips
     *
     * @param int $pValue
     * @return float
     */
    public static function centimetersToTwips($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return $pValue * 566.928;
    }

    /**
     * Convert twips width to centimeters
     *
     * @param int $pValue
     * @return float
     */
    public static function twipsToCentimeters($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return $pValue / 566.928;
    }

    /**
     * Convert inches width to twips
     *
     * @param int $pValue
     * @return float
     */
    public static function inchesToTwips($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return $pValue * 1440;
    }

    /**
     * Convert twips width to inches
     *
     * @param int $pValue
     * @return float
     */
    public static function twipsToInches($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return $pValue / 1440;
    }

    /**
     * Convert twips width to pixels
     *
     * @param int $pValue
     * @return float
     */
    public static function twipsToPixels($pValue = 0)
    {
        if ($pValue == 0) {
            return 0;
        }

        return round($pValue / 15.873984);
    }

    /**
     * Convert HTML hexadecimal to RGB
     *
     * @param string $pValue HTML Color in hexadecimal
     * @return array|false Value in RGB
     */
    public static function htmlToRGB($pValue)
    {
        if ($pValue[0] == '#') {
            $pValue = substr($pValue, 1);
        }

        if (strlen($pValue) == 6) {
            list($colorR, $colorG, $colorB) = array($pValue[0] . $pValue[1], $pValue[2] . $pValue[3], $pValue[4] . $pValue[5]);
        } elseif (strlen($pValue) == 3) {
            list($colorR, $colorG, $colorB) = array($pValue[0] . $pValue[0], $pValue[1] . $pValue[1], $pValue[2] . $pValue[2]);
        } else {
            return false;
        }

        $colorR = hexdec($colorR);
        $colorG = hexdec($colorG);
        $colorB = hexdec($colorB);

        return array($colorR, $colorG, $colorB);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/OLERead.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

use PhpOffice\PhpWord\Exception\Exception;

defined('IDENTIFIER_OLE') ||
define('IDENTIFIER_OLE', pack('CCCCCCCC', 0xd0, 0xcf, 0x11, 0xe0, 0xa1, 0xb1, 0x1a, 0xe1));

class OLERead
{
    private $data = '';

    // OLE identifier
    const IDENTIFIER_OLE = IDENTIFIER_OLE;

    // Size of a sector = 512 bytes
    const BIG_BLOCK_SIZE = 0x200;

    // Size of a short sector = 64 bytes
    const SMALL_BLOCK_SIZE = 0x40;

    // Size of a directory entry always = 128 bytes
    const PROPERTY_STORAGE_BLOCK_SIZE = 0x80;

    // Minimum size of a standard stream = 4096 bytes, streams smaller than this are stored as short streams
    const SMALL_BLOCK_THRESHOLD = 0x1000;

    // header offsets
    const NUM_BIG_BLOCK_DEPOT_BLOCKS_POS = 0x2c;
    const ROOT_START_BLOCK_POS = 0x30;
    const SMALL_BLOCK_DEPOT_BLOCK_POS = 0x3c;
    const EXTENSION_BLOCK_POS = 0x44;
    const NUM_EXTENSION_BLOCK_POS = 0x48;
    const BIG_BLOCK_DEPOT_BLOCKS_POS = 0x4c;

    // property storage offsets (directory offsets)
    const SIZE_OF_NAME_POS = 0x40;
    const TYPE_POS = 0x42;
    const START_BLOCK_POS = 0x74;
    const SIZE_POS = 0x78;

    public $wrkdocument = null;
    public $wrk1Table = null;
    public $wrkData = null;
    public $wrkObjectPool = null;
    public $summaryInformation = null;
    public $docSummaryInfos = null;

    /**
     * Read the file
     *
     * @param $sFileName string Filename
     *
     * @throws Exception
     */
    public function read($sFileName)
    {
        // Check if file exists and is readable
        if (!is_readable($sFileName)) {
            throw new Exception('Could not open ' . $sFileName . ' for reading! File does not exist, or it is not readable.');
        }

        // Get the file identifier
        // Don't bother reading the whole file until we know it's a valid OLE file
        $this->data = file_get_contents($sFileName, false, null, 0, 8);

        // Check OLE identifier
        if ($this->data != self::IDENTIFIER_OLE) {
            throw new Exception('The filename ' . $sFileName . ' is not recognised as an OLE file');
        }

        // Get the file data
        $this->data = file_get_contents($sFileName);

        // Total number of sectors used for the SAT
        $this->numBigBlockDepotBlocks = self::getInt4d($this->data, self::NUM_BIG_BLOCK_DEPOT_BLOCKS_POS);

        // SecID of the first sector of the directory stream
        $this->rootStartBlock = self::getInt4d($this->data, self::ROOT_START_BLOCK_POS);

        // SecID of the first sector of the SSAT (or -2 if not extant)
        $this->sbdStartBlock = self::getInt4d($this->data, self::SMALL_BLOCK_DEPOT_BLOCK_POS);

        // SecID of the first sector of the MSAT (or -2 if no additional sectors are used)
        $this->extensionBlock = self::getInt4d($this->data, self::EXTENSION_BLOCK_POS);

        // Total number of sectors used by MSAT
        $this->numExtensionBlocks = self::getInt4d($this->data, self::NUM_EXTENSION_BLOCK_POS);

        $bigBlockDepotBlocks = array();
        $pos = self::BIG_BLOCK_DEPOT_BLOCKS_POS;

        $bbdBlocks = $this->numBigBlockDepotBlocks;

        // @codeCoverageIgnoreStart
        if ($this->numExtensionBlocks != 0) {
            $bbdBlocks = (self::BIG_BLOCK_SIZE - self::BIG_BLOCK_DEPOT_BLOCKS_POS) / 4;
        }
        // @codeCoverageIgnoreEnd

        for ($i = 0; $i < $bbdBlocks; ++$i) {
            $bigBlockDepotBlocks[$i] = self::getInt4d($this->data, $pos);
            $pos += 4;
        }

        // @codeCoverageIgnoreStart
        for ($j = 0; $j < $this->numExtensionBlocks; ++$j) {
            $pos = ($this->extensionBlock + 1) * self::BIG_BLOCK_SIZE;
            $blocksToRead = min($this->numBigBlockDepotBlocks - $bbdBlocks, self::BIG_BLOCK_SIZE / 4 - 1);

            for ($i = $bbdBlocks; $i < $bbdBlocks + $blocksToRead; ++$i) {
                $bigBlockDepotBlocks[$i] = self::getInt4d($this->data, $pos);
                $pos += 4;
            }

            $bbdBlocks += $blocksToRead;
            if ($bbdBlocks < $this->numBigBlockDepotBlocks) {
                $this->extensionBlock = self::getInt4d($this->data, $pos);
            }
        }
        // @codeCoverageIgnoreEnd

        $pos = 0;
        $this->bigBlockChain = '';
        $bbs = self::BIG_BLOCK_SIZE / 4;
        for ($i = 0; $i < $this->numBigBlockDepotBlocks; ++$i) {
            $pos = ($bigBlockDepotBlocks[$i] + 1) * self::BIG_BLOCK_SIZE;

            $this->bigBlockChain .= substr($this->data, $pos, 4 * $bbs);
            $pos += 4 * $bbs;
        }

        $pos = 0;
        $sbdBlock = $this->sbdStartBlock;
        $this->smallBlockChain = '';
        while ($sbdBlock != -2) {
            $pos = ($sbdBlock + 1) * self::BIG_BLOCK_SIZE;

            $this->smallBlockChain .= substr($this->data, $pos, 4 * $bbs);
            $pos += 4 * $bbs;

            $sbdBlock = self::getInt4d($this->bigBlockChain, $sbdBlock * 4);
        }

        // read the directory stream
        $block = $this->rootStartBlock;
        $this->entry = $this->readData($block);

        $this->readPropertySets();
    }

    /**
     * Extract binary stream data
     *
     * @param mixed $stream
     * @return string
     */
    public function getStream($stream)
    {
        if ($stream === null) {
            return null;
        }

        $streamData = '';

        if ($this->props[$stream]['size'] < self::SMALL_BLOCK_THRESHOLD) {
            $rootdata = $this->readData($this->props[$this->rootentry]['startBlock']);

            $block = $this->props[$stream]['startBlock'];

            while ($block != -2) {
                $pos = $block * self::SMALL_BLOCK_SIZE;
                $streamData .= substr($rootdata, $pos, self::SMALL_BLOCK_SIZE);

                $block = self::getInt4d($this->smallBlockChain, $block * 4);
            }

            return $streamData;
        }

        $numBlocks = $this->props[$stream]['size'] / self::BIG_BLOCK_SIZE;
        if ($this->props[$stream]['size'] % self::BIG_BLOCK_SIZE != 0) {
            ++$numBlocks;
        }

        if ($numBlocks == 0) {
            return ''; // @codeCoverageIgnore
        }

        $block = $this->props[$stream]['startBlock'];

        while ($block != -2) {
            $pos = ($block + 1) * self::BIG_BLOCK_SIZE;
            $streamData .= substr($this->data, $pos, self::BIG_BLOCK_SIZE);
            $block = self::getInt4d($this->bigBlockChain, $block * 4);
        }

        return $streamData;
    }

    /**
     * Read a standard stream (by joining sectors using information from SAT)
     *
     * @param int $blSectorId Sector ID where the stream starts
     * @return string Data for standard stream
     */
    private function readData($blSectorId)
    {
        $block = $blSectorId;
        $data = '';

        while ($block != -2) {
            $pos = ($block + 1) * self::BIG_BLOCK_SIZE;
            $data .= substr($this->data, $pos, self::BIG_BLOCK_SIZE);
            $block = self::getInt4d($this->bigBlockChain, $block * 4);
        }

        return $data;
    }

    /**
     * Read entries in the directory stream.
     */
    private function readPropertySets()
    {
        $offset = 0;

        // loop through entires, each entry is 128 bytes
        $entryLen = strlen($this->entry);
        while ($offset < $entryLen) {
            // entry data (128 bytes)
            $data = substr($this->entry, $offset, self::PROPERTY_STORAGE_BLOCK_SIZE);

            // size in bytes of name
            $nameSize = ord($data[self::SIZE_OF_NAME_POS]) | (ord($data[self::SIZE_OF_NAME_POS + 1]) << 8);

            // type of entry
            $type = ord($data[self::TYPE_POS]);

            // sectorID of first sector or short sector, if this entry refers to a stream (the case with workbook)
            // sectorID of first sector of the short-stream container stream, if this entry is root entry
            $startBlock = self::getInt4d($data, self::START_BLOCK_POS);

            $size = self::getInt4d($data, self::SIZE_POS);

            $name = str_replace("\x00", '', substr($data, 0, $nameSize));

            $this->props[] = array(
                'name'       => $name,
                'type'       => $type,
                'startBlock' => $startBlock,
                'size'       => $size, );

            // tmp helper to simplify checks
            $upName = strtoupper($name);

            // Workbook directory entry (BIFF5 uses Book, BIFF8 uses Workbook)
            // print_r($upName.PHP_EOL);
            if (($upName === 'WORDDOCUMENT')) {
                $this->wrkdocument = count($this->props) - 1;
            } elseif ($upName === '1TABLE') {
                $this->wrk1Table = count($this->props) - 1;
            } elseif ($upName === 'DATA') {
                $this->wrkData = count($this->props) - 1;
            } elseif ($upName === 'OBJECTPOOL') {
                $this->wrkObjectPoolelseif = count($this->props) - 1;
            } elseif ($upName === 'ROOT ENTRY' || $upName === 'R') {
                $this->rootentry = count($this->props) - 1;
            }

            // Summary information
            if ($name == chr(5) . 'SummaryInformation') {
                $this->summaryInformation = count($this->props) - 1;
            }

            // Additional Document Summary information
            if ($name == chr(5) . 'DocumentSummaryInformation') {
                $this->docSummaryInfos = count($this->props) - 1;
            }

            $offset += self::PROPERTY_STORAGE_BLOCK_SIZE;
        }
    }

    /**
     * Read 4 bytes of data at specified position
     *
     * @param string $data
     * @param int $pos
     * @return int
     */
    private static function getInt4d($data, $pos)
    {
        // FIX: represent numbers correctly on 64-bit system
        // http://sourceforge.net/tracker/index.php?func=detail&aid=1487372&group_id=99160&atid=623334
        // Hacked by Andreas Rehm 2006 to ensure correct result of the <<24 block on 32 and 64bit systems
        $or24 = ord($data[$pos + 3]);
        if ($or24 >= 128) {
            // negative number
            $ord24 = -abs((256 - $or24) << 24);
        } else {
            $ord24 = ($or24 & 127) << 24;
        }

        return ord($data[$pos]) | (ord($data[$pos + 1]) << 8) | (ord($data[$pos + 2]) << 16) | $ord24;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/PCLZip/pclzip.lib.php

<?php
// --------------------------------------------------------------------------------
// PhpConcept Library - Zip Module 2.8.2
// --------------------------------------------------------------------------------
// License GNU/LGPL - Vincent Blavet - August 2009
// http://www.phpconcept.net
// --------------------------------------------------------------------------------
//
// Presentation :
//   PclZip is a PHP library that manage ZIP archives.
//   So far tests show that archives generated by PclZip are readable by
//   WinZip application and other tools.
//
// Description :
//   See readme.txt and http://www.phpconcept.net
//
// Warning :
//   This library and the associated files are non commercial, non professional
//   work.
//   It should not have unexpected results. However if any damage is caused by
//   this software the author can not be responsible.
//   The use of this software is at the risk of the user.
//
// --------------------------------------------------------------------------------
// $Id: pclzip.lib.php,v 1.60 2009/09/30 21:01:04 vblavet Exp $
// --------------------------------------------------------------------------------

// ----- Constants
if (!defined('PCLZIP_READ_BLOCK_SIZE')) {
    define('PCLZIP_READ_BLOCK_SIZE', 2048);
}

// ----- File list separator
// In version 1.x of PclZip, the separator for file list is a space
// (which is not a very smart choice, specifically for windows paths !).
// A better separator should be a comma (,). This constant gives you the
// abilty to change that.
// However notice that changing this value, may have impact on existing
// scripts, using space separated filenames.
// Recommanded values for compatibility with older versions :
//define( 'PCLZIP_SEPARATOR', ' ' );
// Recommanded values for smart separation of filenames.
if (!defined('PCLZIP_SEPARATOR')) {
    define('PCLZIP_SEPARATOR', ',');
}

// ----- Error configuration
// 0 : PclZip Class integrated error handling
// 1 : PclError external library error handling. By enabling this
//     you must ensure that you have included PclError library.
// [2,...] : reserved for futur use
if (!defined('PCLZIP_ERROR_EXTERNAL')) {
    define('PCLZIP_ERROR_EXTERNAL', 0);
}

// ----- Optional static temporary directory
//       By default temporary files are generated in the script current
//       path.
//       If defined :
//       - MUST BE terminated by a '/'.
//       - MUST be a valid, already created directory
//       Samples :
// define( 'PCLZIP_TEMPORARY_DIR', '/temp/' );
// define( 'PCLZIP_TEMPORARY_DIR', 'C:/Temp/' );
if (!defined('PCLZIP_TEMPORARY_DIR')) {
    define('PCLZIP_TEMPORARY_DIR', '');
}

// ----- Optional threshold ratio for use of temporary files
//       Pclzip sense the size of the file to add/extract and decide to
//       use or not temporary file. The algorythm is looking for
//       memory_limit of PHP and apply a ratio.
//       threshold = memory_limit * ratio.
//       Recommended values are under 0.5. Default 0.47.
//       Samples :
// define( 'PCLZIP_TEMPORARY_FILE_RATIO', 0.5 );
if (!defined('PCLZIP_TEMPORARY_FILE_RATIO')) {
    define('PCLZIP_TEMPORARY_FILE_RATIO', 0.47);
}

// --------------------------------------------------------------------------------
// ***** UNDER THIS LINE NOTHING NEEDS TO BE MODIFIED *****
// --------------------------------------------------------------------------------

// ----- Global variables
$g_pclzip_version = "2.8.2";

// ----- Error codes
//   -1 : Unable to open file in binary write mode
//   -2 : Unable to open file in binary read mode
//   -3 : Invalid parameters
//   -4 : File does not exist
//   -5 : Filename is too long (max. 255)
//   -6 : Not a valid zip file
//   -7 : Invalid extracted file size
//   -8 : Unable to create directory
//   -9 : Invalid archive extension
//  -10 : Invalid archive format
//  -11 : Unable to delete file (unlink)
//  -12 : Unable to rename file (rename)
//  -13 : Invalid header checksum
//  -14 : Invalid archive size
define('PCLZIP_ERR_USER_ABORTED', 2);
define('PCLZIP_ERR_NO_ERROR', 0);
define('PCLZIP_ERR_WRITE_OPEN_FAIL', -1);
define('PCLZIP_ERR_READ_OPEN_FAIL', -2);
define('PCLZIP_ERR_INVALID_PARAMETER', -3);
define('PCLZIP_ERR_MISSING_FILE', -4);
define('PCLZIP_ERR_FILENAME_TOO_LONG', -5);
define('PCLZIP_ERR_INVALID_ZIP', -6);
define('PCLZIP_ERR_BAD_EXTRACTED_FILE', -7);
define('PCLZIP_ERR_DIR_CREATE_FAIL', -8);
define('PCLZIP_ERR_BAD_EXTENSION', -9);
define('PCLZIP_ERR_BAD_FORMAT', -10);
define('PCLZIP_ERR_DELETE_FILE_FAIL', -11);
define('PCLZIP_ERR_RENAME_FILE_FAIL', -12);
define('PCLZIP_ERR_BAD_CHECKSUM', -13);
define('PCLZIP_ERR_INVALID_ARCHIVE_ZIP', -14);
define('PCLZIP_ERR_MISSING_OPTION_VALUE', -15);
define('PCLZIP_ERR_INVALID_OPTION_VALUE', -16);
define('PCLZIP_ERR_ALREADY_A_DIRECTORY', -17);
define('PCLZIP_ERR_UNSUPPORTED_COMPRESSION', -18);
define('PCLZIP_ERR_UNSUPPORTED_ENCRYPTION', -19);
define('PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE', -20);
define('PCLZIP_ERR_DIRECTORY_RESTRICTION', -21);

// ----- Options values
define('PCLZIP_OPT_PATH', 77001);
define('PCLZIP_OPT_ADD_PATH', 77002);
define('PCLZIP_OPT_REMOVE_PATH', 77003);
define('PCLZIP_OPT_REMOVE_ALL_PATH', 77004);
define('PCLZIP_OPT_SET_CHMOD', 77005);
define('PCLZIP_OPT_EXTRACT_AS_STRING', 77006);
define('PCLZIP_OPT_NO_COMPRESSION', 77007);
define('PCLZIP_OPT_BY_NAME', 77008);
define('PCLZIP_OPT_BY_INDEX', 77009);
define('PCLZIP_OPT_BY_EREG', 77010);
define('PCLZIP_OPT_BY_PREG', 77011);
define('PCLZIP_OPT_COMMENT', 77012);
define('PCLZIP_OPT_ADD_COMMENT', 77013);
define('PCLZIP_OPT_PREPEND_COMMENT', 77014);
define('PCLZIP_OPT_EXTRACT_IN_OUTPUT', 77015);
define('PCLZIP_OPT_REPLACE_NEWER', 77016);
define('PCLZIP_OPT_STOP_ON_ERROR', 77017);
// Having big trouble with crypt. Need to multiply 2 long int
// which is not correctly supported by PHP ...
//define( 'PCLZIP_OPT_CRYPT', 77018 );
define('PCLZIP_OPT_EXTRACT_DIR_RESTRICTION', 77019);
define('PCLZIP_OPT_TEMP_FILE_THRESHOLD', 77020);
define('PCLZIP_OPT_ADD_TEMP_FILE_THRESHOLD', 77020); // alias
define('PCLZIP_OPT_TEMP_FILE_ON', 77021);
define('PCLZIP_OPT_ADD_TEMP_FILE_ON', 77021); // alias
define('PCLZIP_OPT_TEMP_FILE_OFF', 77022);
define('PCLZIP_OPT_ADD_TEMP_FILE_OFF', 77022); // alias

// ----- File description attributes
define('PCLZIP_ATT_FILE_NAME', 79001);
define('PCLZIP_ATT_FILE_NEW_SHORT_NAME', 79002);
define('PCLZIP_ATT_FILE_NEW_FULL_NAME', 79003);
define('PCLZIP_ATT_FILE_MTIME', 79004);
define('PCLZIP_ATT_FILE_CONTENT', 79005);
define('PCLZIP_ATT_FILE_COMMENT', 79006);

// ----- Call backs values
define('PCLZIP_CB_PRE_EXTRACT', 78001);
define('PCLZIP_CB_POST_EXTRACT', 78002);
define('PCLZIP_CB_PRE_ADD', 78003);
define('PCLZIP_CB_POST_ADD', 78004);
/* For futur use
define( 'PCLZIP_CB_PRE_LIST', 78005 );
define( 'PCLZIP_CB_POST_LIST', 78006 );
define( 'PCLZIP_CB_PRE_DELETE', 78007 );
define( 'PCLZIP_CB_POST_DELETE', 78008 );
*/

// --------------------------------------------------------------------------------
// Class : PclZip
// Description :
//   PclZip is the class that represent a Zip archive.
//   The public methods allow the manipulation of the archive.
// Attributes :
//   Attributes must not be accessed directly.
// Methods :
//   PclZip() : Object creator
//   create() : Creates the Zip archive
//   listContent() : List the content of the Zip archive
//   extract() : Extract the content of the archive
//   properties() : List the properties of the archive
// --------------------------------------------------------------------------------
class PclZip
{
    // ----- Filename of the zip file
    public $zipname = '';

    // ----- File descriptor of the zip file
    public $zip_fd = 0;

    // ----- Internal error handling
    public $error_code = 1;
    public $error_string = '';

    // ----- Current status of the magic_quotes_runtime
    // This value store the php configuration for magic_quotes
    // The class can then disable the magic_quotes and reset it after
    public $magic_quotes_status;

    // --------------------------------------------------------------------------------
    // Function : PclZip()
    // Description :
    //   Creates a PclZip object and set the name of the associated Zip archive
    //   filename.
    //   Note that no real action is taken, if the archive does not exist it is not
    //   created. Use create() for that.
    // --------------------------------------------------------------------------------
    public function __construct($p_zipname)
    {

        // ----- Tests the zlib
        if (!function_exists('gzopen')) {
            die('Abort ' . basename(__FILE__) . ' : Missing zlib extensions');
        }

        // ----- Set the attributes
        $this->zipname             = $p_zipname;
        $this->zip_fd              = 0;
        $this->magic_quotes_status = -1;

        // ----- Return
        return;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function :
    //   create($p_filelist, $p_add_dir="", $p_remove_dir="")
    //   create($p_filelist, $p_option, $p_option_value, ...)
    // Description :
    //   This method supports two different synopsis. The first one is historical.
    //   This method creates a Zip Archive. The Zip file is created in the
    //   filesystem. The files and directories indicated in $p_filelist
    //   are added in the archive. See the parameters description for the
    //   supported format of $p_filelist.
    //   When a directory is in the list, the directory and its content is added
    //   in the archive.
    //   In this synopsis, the function takes an optional variable list of
    //   options. See bellow the supported options.
    // Parameters :
    //   $p_filelist : An array containing file or directory names, or
    //                 a string containing one filename or one directory name, or
    //                 a string containing a list of filenames and/or directory
    //                 names separated by spaces.
    //   $p_add_dir : A path to add before the real path of the archived file,
    //                in order to have it memorized in the archive.
    //   $p_remove_dir : A path to remove from the real path of the file to archive,
    //                   in order to have a shorter path memorized in the archive.
    //                   When $p_add_dir and $p_remove_dir are set, $p_remove_dir
    //                   is removed first, before $p_add_dir is added.
    // Options :
    //   PCLZIP_OPT_ADD_PATH :
    //   PCLZIP_OPT_REMOVE_PATH :
    //   PCLZIP_OPT_REMOVE_ALL_PATH :
    //   PCLZIP_OPT_COMMENT :
    //   PCLZIP_CB_PRE_ADD :
    //   PCLZIP_CB_POST_ADD :
    // Return Values :
    //   0 on failure,
    //   The list of the added files, with a status of the add action.
    //   (see PclZip::listContent() for list entry format)
    // --------------------------------------------------------------------------------
    public function create($p_filelist)
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Set default values
        $v_options                            = array();
        $v_options[PCLZIP_OPT_NO_COMPRESSION] = false;

        // ----- Look for variable options arguments
        $v_size = func_num_args();

        // ----- Look for arguments
        if ($v_size > 1) {
            // ----- Get the arguments
            $v_arg_list = func_get_args();

            // ----- Remove from the options list the first argument
            array_shift($v_arg_list);
            $v_size--;

            // ----- Look for first arg
            if ((is_integer($v_arg_list[0])) && ($v_arg_list[0] > 77000)) {

                // ----- Parse the options
                $v_result = $this->privParseOptions($v_arg_list, $v_size, $v_options, array(
                    PCLZIP_OPT_REMOVE_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_ALL_PATH => 'optional',
                    PCLZIP_OPT_ADD_PATH => 'optional',
                    PCLZIP_CB_PRE_ADD => 'optional',
                    PCLZIP_CB_POST_ADD => 'optional',
                    PCLZIP_OPT_NO_COMPRESSION => 'optional',
                    PCLZIP_OPT_COMMENT => 'optional',
                    PCLZIP_OPT_TEMP_FILE_THRESHOLD => 'optional',
                    PCLZIP_OPT_TEMP_FILE_ON => 'optional',
                    PCLZIP_OPT_TEMP_FILE_OFF => 'optional'
                    //, PCLZIP_OPT_CRYPT => 'optional'
                ));
                if ($v_result != 1) {
                    return 0;
                }

            // ----- Look for 2 args
            // Here we need to support the first historic synopsis of the
            // method.
            } else {

                // ----- Get the first argument
                $v_options[PCLZIP_OPT_ADD_PATH] = $v_arg_list[0];

                // ----- Look for the optional second argument
                if ($v_size == 2) {
                    $v_options[PCLZIP_OPT_REMOVE_PATH] = $v_arg_list[1];
                } elseif ($v_size > 2) {
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid number / type of arguments");

                    return 0;
                }
            }
        }

        // ----- Look for default option values
        $this->privOptionDefaultThreshold($v_options);

        // ----- Init
        $v_string_list    = array();
        $v_att_list       = array();
        $v_filedescr_list = array();
        $p_result_list    = array();

        // ----- Look if the $p_filelist is really an array
        if (is_array($p_filelist)) {

            // ----- Look if the first element is also an array
            //       This will mean that this is a file description entry
            if (isset($p_filelist[0]) && is_array($p_filelist[0])) {
                $v_att_list = $p_filelist;

            // ----- The list is a list of string names
            } else {
                $v_string_list = $p_filelist;
            }

        // ----- Look if the $p_filelist is a string
        } elseif (is_string($p_filelist)) {
            // ----- Create a list from the string
            $v_string_list = explode(PCLZIP_SEPARATOR, $p_filelist);

        // ----- Invalid variable type for $p_filelist
        } else {
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid variable type p_filelist");

            return 0;
        }

        // ----- Reformat the string list
        if (sizeof($v_string_list) != 0) {
            foreach ($v_string_list as $v_string) {
                if ($v_string != '') {
                    $v_att_list[][PCLZIP_ATT_FILE_NAME] = $v_string;
                } else {
                }
            }
        }

        // ----- For each file in the list check the attributes
        $v_supported_attributes = array(
            PCLZIP_ATT_FILE_NAME => 'mandatory',
            PCLZIP_ATT_FILE_NEW_SHORT_NAME => 'optional',
            PCLZIP_ATT_FILE_NEW_FULL_NAME => 'optional',
            PCLZIP_ATT_FILE_MTIME => 'optional',
            PCLZIP_ATT_FILE_CONTENT => 'optional',
            PCLZIP_ATT_FILE_COMMENT => 'optional'
        );
        foreach ($v_att_list as $v_entry) {
            $v_result = $this->privFileDescrParseAtt($v_entry, $v_filedescr_list[], $v_options, $v_supported_attributes);
            if ($v_result != 1) {
                return 0;
            }
        }

        // ----- Expand the filelist (expand directories)
        $v_result = $this->privFileDescrExpand($v_filedescr_list, $v_options);
        if ($v_result != 1) {
            return 0;
        }

        // ----- Call the create fct
        $v_result = $this->privCreate($v_filedescr_list, $p_result_list, $v_options);
        if ($v_result != 1) {
            return 0;
        }

        // ----- Return
        return $p_result_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function :
    //   add($p_filelist, $p_add_dir="", $p_remove_dir="")
    //   add($p_filelist, $p_option, $p_option_value, ...)
    // Description :
    //   This method supports two synopsis. The first one is historical.
    //   This methods add the list of files in an existing archive.
    //   If a file with the same name already exists, it is added at the end of the
    //   archive, the first one is still present.
    //   If the archive does not exist, it is created.
    // Parameters :
    //   $p_filelist : An array containing file or directory names, or
    //                 a string containing one filename or one directory name, or
    //                 a string containing a list of filenames and/or directory
    //                 names separated by spaces.
    //   $p_add_dir : A path to add before the real path of the archived file,
    //                in order to have it memorized in the archive.
    //   $p_remove_dir : A path to remove from the real path of the file to archive,
    //                   in order to have a shorter path memorized in the archive.
    //                   When $p_add_dir and $p_remove_dir are set, $p_remove_dir
    //                   is removed first, before $p_add_dir is added.
    // Options :
    //   PCLZIP_OPT_ADD_PATH :
    //   PCLZIP_OPT_REMOVE_PATH :
    //   PCLZIP_OPT_REMOVE_ALL_PATH :
    //   PCLZIP_OPT_COMMENT :
    //   PCLZIP_OPT_ADD_COMMENT :
    //   PCLZIP_OPT_PREPEND_COMMENT :
    //   PCLZIP_CB_PRE_ADD :
    //   PCLZIP_CB_POST_ADD :
    // Return Values :
    //   0 on failure,
    //   The list of the added files, with a status of the add action.
    //   (see PclZip::listContent() for list entry format)
    // --------------------------------------------------------------------------------
    public function add($p_filelist)
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Set default values
        $v_options                            = array();
        $v_options[PCLZIP_OPT_NO_COMPRESSION] = false;

        // ----- Look for variable options arguments
        $v_size = func_num_args();

        // ----- Look for arguments
        if ($v_size > 1) {
            // ----- Get the arguments
            $v_arg_list = func_get_args();

            // ----- Remove form the options list the first argument
            array_shift($v_arg_list);
            $v_size--;

            // ----- Look for first arg
            if ((is_integer($v_arg_list[0])) && ($v_arg_list[0] > 77000)) {

                // ----- Parse the options
                $v_result = $this->privParseOptions($v_arg_list, $v_size, $v_options, array(
                    PCLZIP_OPT_REMOVE_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_ALL_PATH => 'optional',
                    PCLZIP_OPT_ADD_PATH => 'optional',
                    PCLZIP_CB_PRE_ADD => 'optional',
                    PCLZIP_CB_POST_ADD => 'optional',
                    PCLZIP_OPT_NO_COMPRESSION => 'optional',
                    PCLZIP_OPT_COMMENT => 'optional',
                    PCLZIP_OPT_ADD_COMMENT => 'optional',
                    PCLZIP_OPT_PREPEND_COMMENT => 'optional',
                    PCLZIP_OPT_TEMP_FILE_THRESHOLD => 'optional',
                    PCLZIP_OPT_TEMP_FILE_ON => 'optional',
                    PCLZIP_OPT_TEMP_FILE_OFF => 'optional'
                    //, PCLZIP_OPT_CRYPT => 'optional'
                ));
                if ($v_result != 1) {
                    return 0;
                }

            // ----- Look for 2 args
            // Here we need to support the first historic synopsis of the
            // method.
            } else {

                // ----- Get the first argument
                $v_options[PCLZIP_OPT_ADD_PATH] = $v_add_path = $v_arg_list[0];

                // ----- Look for the optional second argument
                if ($v_size == 2) {
                    $v_options[PCLZIP_OPT_REMOVE_PATH] = $v_arg_list[1];
                } elseif ($v_size > 2) {
                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid number / type of arguments");

                    // ----- Return
                    return 0;
                }
            }
        }

        // ----- Look for default option values
        $this->privOptionDefaultThreshold($v_options);

        // ----- Init
        $v_string_list    = array();
        $v_att_list       = array();
        $v_filedescr_list = array();
        $p_result_list    = array();

        // ----- Look if the $p_filelist is really an array
        if (is_array($p_filelist)) {

            // ----- Look if the first element is also an array
            //       This will mean that this is a file description entry
            if (isset($p_filelist[0]) && is_array($p_filelist[0])) {
                $v_att_list = $p_filelist;

            // ----- The list is a list of string names
            } else {
                $v_string_list = $p_filelist;
            }

        // ----- Look if the $p_filelist is a string
        } elseif (is_string($p_filelist)) {
            // ----- Create a list from the string
            $v_string_list = explode(PCLZIP_SEPARATOR, $p_filelist);

        // ----- Invalid variable type for $p_filelist
        } else {
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid variable type '" . gettype($p_filelist) . "' for p_filelist");

            return 0;
        }

        // ----- Reformat the string list
        if (sizeof($v_string_list) != 0) {
            foreach ($v_string_list as $v_string) {
                $v_att_list[][PCLZIP_ATT_FILE_NAME] = $v_string;
            }
        }

        // ----- For each file in the list check the attributes
        $v_supported_attributes = array(
            PCLZIP_ATT_FILE_NAME => 'mandatory',
            PCLZIP_ATT_FILE_NEW_SHORT_NAME => 'optional',
            PCLZIP_ATT_FILE_NEW_FULL_NAME => 'optional',
            PCLZIP_ATT_FILE_MTIME => 'optional',
            PCLZIP_ATT_FILE_CONTENT => 'optional',
            PCLZIP_ATT_FILE_COMMENT => 'optional'
        );
        foreach ($v_att_list as $v_entry) {
            $v_result = $this->privFileDescrParseAtt($v_entry, $v_filedescr_list[], $v_options, $v_supported_attributes);
            if ($v_result != 1) {
                return 0;
            }
        }

        // ----- Expand the filelist (expand directories)
        $v_result = $this->privFileDescrExpand($v_filedescr_list, $v_options);
        if ($v_result != 1) {
            return 0;
        }

        // ----- Call the create fct
        $v_result = $this->privAdd($v_filedescr_list, $p_result_list, $v_options);
        if ($v_result != 1) {
            return 0;
        }

        // ----- Return
        return $p_result_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : listContent()
    // Description :
    //   This public method, gives the list of the files and directories, with their
    //   properties.
    //   The properties of each entries in the list are (used also in other functions) :
    //     filename : Name of the file. For a create or add action it is the filename
    //                given by the user. For an extract function it is the filename
    //                of the extracted file.
    //     stored_filename : Name of the file / directory stored in the archive.
    //     size : Size of the stored file.
    //     compressed_size : Size of the file's data compressed in the archive
    //                       (without the headers overhead)
    //     mtime : Last known modification date of the file (UNIX timestamp)
    //     comment : Comment associated with the file
    //     folder : true | false
    //     index : index of the file in the archive
    //     status : status of the action (depending of the action) :
    //              Values are :
    //                ok : OK !
    //                filtered : the file / dir is not extracted (filtered by user)
    //                already_a_directory : the file can not be extracted because a
    //                                      directory with the same name already exists
    //                write_protected : the file can not be extracted because a file
    //                                  with the same name already exists and is
    //                                  write protected
    //                newer_exist : the file was not extracted because a newer file exists
    //                path_creation_fail : the file is not extracted because the folder
    //                                     does not exist and can not be created
    //                write_error : the file was not extracted because there was a
    //                              error while writing the file
    //                read_error : the file was not extracted because there was a error
    //                             while reading the file
    //                invalid_header : the file was not extracted because of an archive
    //                                 format error (bad file header)
    //   Note that each time a method can continue operating when there
    //   is an action error on a file, the error is only logged in the file status.
    // Return Values :
    //   0 on an unrecoverable failure,
    //   The list of the files in the archive.
    // --------------------------------------------------------------------------------
    public function listContent()
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            return (0);
        }

        // ----- Call the extracting fct
        $p_list = array();
        if (($v_result = $this->privList($p_list)) != 1) {
            unset($p_list);

            return (0);
        }

        // ----- Return
        return $p_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function :
    //   extract($p_path="./", $p_remove_path="")
    //   extract([$p_option, $p_option_value, ...])
    // Description :
    //   This method supports two synopsis. The first one is historical.
    //   This method extract all the files / directories from the archive to the
    //   folder indicated in $p_path.
    //   If you want to ignore the 'root' part of path of the memorized files
    //   you can indicate this in the optional $p_remove_path parameter.
    //   By default, if a newer file with the same name already exists, the
    //   file is not extracted.
    //
    //   If both PCLZIP_OPT_PATH and PCLZIP_OPT_ADD_PATH aoptions
    //   are used, the path indicated in PCLZIP_OPT_ADD_PATH is append
    //   at the end of the path value of PCLZIP_OPT_PATH.
    // Parameters :
    //   $p_path : Path where the files and directories are to be extracted
    //   $p_remove_path : First part ('root' part) of the memorized path
    //                    (if any similar) to remove while extracting.
    // Options :
    //   PCLZIP_OPT_PATH :
    //   PCLZIP_OPT_ADD_PATH :
    //   PCLZIP_OPT_REMOVE_PATH :
    //   PCLZIP_OPT_REMOVE_ALL_PATH :
    //   PCLZIP_CB_PRE_EXTRACT :
    //   PCLZIP_CB_POST_EXTRACT :
    // Return Values :
    //   0 or a negative value on failure,
    //   The list of the extracted files, with a status of the action.
    //   (see PclZip::listContent() for list entry format)
    // --------------------------------------------------------------------------------
    public function extract()
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            return (0);
        }

        // ----- Set default values
        $v_options         = array();
        //    $v_path = "./";
        $v_path            = '';
        $v_remove_path     = "";
        $v_remove_all_path = false;

        // ----- Look for variable options arguments
        $v_size = func_num_args();

        // ----- Default values for option
        $v_options[PCLZIP_OPT_EXTRACT_AS_STRING] = false;

        // ----- Look for arguments
        if ($v_size > 0) {
            // ----- Get the arguments
            $v_arg_list = func_get_args();

            // ----- Look for first arg
            if ((is_integer($v_arg_list[0])) && ($v_arg_list[0] > 77000)) {

                // ----- Parse the options
                $v_result = $this->privParseOptions($v_arg_list, $v_size, $v_options, array(
                    PCLZIP_OPT_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_ALL_PATH => 'optional',
                    PCLZIP_OPT_ADD_PATH => 'optional',
                    PCLZIP_CB_PRE_EXTRACT => 'optional',
                    PCLZIP_CB_POST_EXTRACT => 'optional',
                    PCLZIP_OPT_SET_CHMOD => 'optional',
                    PCLZIP_OPT_BY_NAME => 'optional',
                    PCLZIP_OPT_BY_EREG => 'optional',
                    PCLZIP_OPT_BY_PREG => 'optional',
                    PCLZIP_OPT_BY_INDEX => 'optional',
                    PCLZIP_OPT_EXTRACT_AS_STRING => 'optional',
                    PCLZIP_OPT_EXTRACT_IN_OUTPUT => 'optional',
                    PCLZIP_OPT_REPLACE_NEWER => 'optional',
                    PCLZIP_OPT_STOP_ON_ERROR => 'optional',
                    PCLZIP_OPT_EXTRACT_DIR_RESTRICTION => 'optional',
                    PCLZIP_OPT_TEMP_FILE_THRESHOLD => 'optional',
                    PCLZIP_OPT_TEMP_FILE_ON => 'optional',
                    PCLZIP_OPT_TEMP_FILE_OFF => 'optional'
                ));
                if ($v_result != 1) {
                    return 0;
                }

                // ----- Set the arguments
                if (isset($v_options[PCLZIP_OPT_PATH])) {
                    $v_path = $v_options[PCLZIP_OPT_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_REMOVE_PATH])) {
                    $v_remove_path = $v_options[PCLZIP_OPT_REMOVE_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_REMOVE_ALL_PATH])) {
                    $v_remove_all_path = $v_options[PCLZIP_OPT_REMOVE_ALL_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_ADD_PATH])) {
                    // ----- Check for '/' in last path char
                    if ((strlen($v_path) > 0) && (substr($v_path, -1) != '/')) {
                        $v_path .= '/';
                    }
                    $v_path .= $v_options[PCLZIP_OPT_ADD_PATH];
                }

            // ----- Look for 2 args
            // Here we need to support the first historic synopsis of the
            // method.
            } else {

                // ----- Get the first argument
                $v_path = $v_arg_list[0];

                // ----- Look for the optional second argument
                if ($v_size == 2) {
                    $v_remove_path = $v_arg_list[1];
                } elseif ($v_size > 2) {
                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid number / type of arguments");

                    // ----- Return
                    return 0;
                }
            }
        }

        // ----- Look for default option values
        $this->privOptionDefaultThreshold($v_options);

        // ----- Trace

        // ----- Call the extracting fct
        $p_list   = array();
        $v_result = $this->privExtractByRule($p_list, $v_path, $v_remove_path, $v_remove_all_path, $v_options);
        if ($v_result < 1) {
            unset($p_list);

            return (0);
        }

        // ----- Return
        return $p_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function :
    //   extractByIndex($p_index, $p_path="./", $p_remove_path="")
    //   extractByIndex($p_index, [$p_option, $p_option_value, ...])
    // Description :
    //   This method supports two synopsis. The first one is historical.
    //   This method is doing a partial extract of the archive.
    //   The extracted files or folders are identified by their index in the
    //   archive (from 0 to n).
    //   Note that if the index identify a folder, only the folder entry is
    //   extracted, not all the files included in the archive.
    // Parameters :
    //   $p_index : A single index (integer) or a string of indexes of files to
    //              extract. The form of the string is "0,4-6,8-12" with only numbers
    //              and '-' for range or ',' to separate ranges. No spaces or ';'
    //              are allowed.
    //   $p_path : Path where the files and directories are to be extracted
    //   $p_remove_path : First part ('root' part) of the memorized path
    //                    (if any similar) to remove while extracting.
    // Options :
    //   PCLZIP_OPT_PATH :
    //   PCLZIP_OPT_ADD_PATH :
    //   PCLZIP_OPT_REMOVE_PATH :
    //   PCLZIP_OPT_REMOVE_ALL_PATH :
    //   PCLZIP_OPT_EXTRACT_AS_STRING : The files are extracted as strings and
    //     not as files.
    //     The resulting content is in a new field 'content' in the file
    //     structure.
    //     This option must be used alone (any other options are ignored).
    //   PCLZIP_CB_PRE_EXTRACT :
    //   PCLZIP_CB_POST_EXTRACT :
    // Return Values :
    //   0 on failure,
    //   The list of the extracted files, with a status of the action.
    //   (see PclZip::listContent() for list entry format)
    // --------------------------------------------------------------------------------
    //function extractByIndex($p_index, options...)
    public function extractByIndex($p_index)
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            return (0);
        }

        // ----- Set default values
        $v_options         = array();
        //    $v_path = "./";
        $v_path            = '';
        $v_remove_path     = "";
        $v_remove_all_path = false;

        // ----- Look for variable options arguments
        $v_size = func_num_args();

        // ----- Default values for option
        $v_options[PCLZIP_OPT_EXTRACT_AS_STRING] = false;

        // ----- Look for arguments
        if ($v_size > 1) {
            // ----- Get the arguments
            $v_arg_list = func_get_args();

            // ----- Remove form the options list the first argument
            array_shift($v_arg_list);
            $v_size--;

            // ----- Look for first arg
            if ((is_integer($v_arg_list[0])) && ($v_arg_list[0] > 77000)) {

                // ----- Parse the options
                $v_result = $this->privParseOptions($v_arg_list, $v_size, $v_options, array(
                    PCLZIP_OPT_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_PATH => 'optional',
                    PCLZIP_OPT_REMOVE_ALL_PATH => 'optional',
                    PCLZIP_OPT_EXTRACT_AS_STRING => 'optional',
                    PCLZIP_OPT_ADD_PATH => 'optional',
                    PCLZIP_CB_PRE_EXTRACT => 'optional',
                    PCLZIP_CB_POST_EXTRACT => 'optional',
                    PCLZIP_OPT_SET_CHMOD => 'optional',
                    PCLZIP_OPT_REPLACE_NEWER => 'optional',
                    PCLZIP_OPT_STOP_ON_ERROR => 'optional',
                    PCLZIP_OPT_EXTRACT_DIR_RESTRICTION => 'optional',
                    PCLZIP_OPT_TEMP_FILE_THRESHOLD => 'optional',
                    PCLZIP_OPT_TEMP_FILE_ON => 'optional',
                    PCLZIP_OPT_TEMP_FILE_OFF => 'optional'
                ));
                if ($v_result != 1) {
                    return 0;
                }

                // ----- Set the arguments
                if (isset($v_options[PCLZIP_OPT_PATH])) {
                    $v_path = $v_options[PCLZIP_OPT_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_REMOVE_PATH])) {
                    $v_remove_path = $v_options[PCLZIP_OPT_REMOVE_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_REMOVE_ALL_PATH])) {
                    $v_remove_all_path = $v_options[PCLZIP_OPT_REMOVE_ALL_PATH];
                }
                if (isset($v_options[PCLZIP_OPT_ADD_PATH])) {
                    // ----- Check for '/' in last path char
                    if ((strlen($v_path) > 0) && (substr($v_path, -1) != '/')) {
                        $v_path .= '/';
                    }
                    $v_path .= $v_options[PCLZIP_OPT_ADD_PATH];
                }
                if (!isset($v_options[PCLZIP_OPT_EXTRACT_AS_STRING])) {
                    $v_options[PCLZIP_OPT_EXTRACT_AS_STRING] = false;
                } else {
                }

            // ----- Look for 2 args
            // Here we need to support the first historic synopsis of the
            // method.
            } else {

                // ----- Get the first argument
                $v_path = $v_arg_list[0];

                // ----- Look for the optional second argument
                if ($v_size == 2) {
                    $v_remove_path = $v_arg_list[1];
                } elseif ($v_size > 2) {
                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid number / type of arguments");

                    // ----- Return
                    return 0;
                }
            }
        }

        // ----- Trace

        // ----- Trick
        // Here I want to reuse extractByRule(), so I need to parse the $p_index
        // with privParseOptions()
        $v_arg_trick     = array(
            PCLZIP_OPT_BY_INDEX,
            $p_index
        );
        $v_options_trick = array();
        $v_result        = $this->privParseOptions($v_arg_trick, sizeof($v_arg_trick), $v_options_trick, array(
            PCLZIP_OPT_BY_INDEX => 'optional'
        ));
        if ($v_result != 1) {
            return 0;
        }
        $v_options[PCLZIP_OPT_BY_INDEX] = $v_options_trick[PCLZIP_OPT_BY_INDEX];

        // ----- Look for default option values
        $this->privOptionDefaultThreshold($v_options);

        // ----- Call the extracting fct
        if (($v_result = $this->privExtractByRule($p_list, $v_path, $v_remove_path, $v_remove_all_path, $v_options)) < 1) {
            return (0);
        }

        // ----- Return
        return $p_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function :
    //   delete([$p_option, $p_option_value, ...])
    // Description :
    //   This method removes files from the archive.
    //   If no parameters are given, then all the archive is emptied.
    // Parameters :
    //   None or optional arguments.
    // Options :
    //   PCLZIP_OPT_BY_INDEX :
    //   PCLZIP_OPT_BY_NAME :
    //   PCLZIP_OPT_BY_EREG :
    //   PCLZIP_OPT_BY_PREG :
    // Return Values :
    //   0 on failure,
    //   The list of the files which are still present in the archive.
    //   (see PclZip::listContent() for list entry format)
    // --------------------------------------------------------------------------------
    public function delete()
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            return (0);
        }

        // ----- Set default values
        $v_options = array();

        // ----- Look for variable options arguments
        $v_size = func_num_args();

        // ----- Look for arguments
        if ($v_size > 0) {
            // ----- Get the arguments
            $v_arg_list = func_get_args();

            // ----- Parse the options
            $v_result = $this->privParseOptions($v_arg_list, $v_size, $v_options, array(
                PCLZIP_OPT_BY_NAME => 'optional',
                PCLZIP_OPT_BY_EREG => 'optional',
                PCLZIP_OPT_BY_PREG => 'optional',
                PCLZIP_OPT_BY_INDEX => 'optional'
            ));
            if ($v_result != 1) {
                return 0;
            }
        }

        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Call the delete fct
        $v_list = array();
        if (($v_result = $this->privDeleteByRule($v_list, $v_options)) != 1) {
            $this->privSwapBackMagicQuotes();
            unset($v_list);

            return (0);
        }

        // ----- Magic quotes trick
        $this->privSwapBackMagicQuotes();

        // ----- Return
        return $v_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : deleteByIndex()
    // Description :
    //   ***** Deprecated *****
    //   delete(PCLZIP_OPT_BY_INDEX, $p_index) should be prefered.
    // --------------------------------------------------------------------------------
    public function deleteByIndex($p_index)
    {

        $p_list = $this->delete(PCLZIP_OPT_BY_INDEX, $p_index);

        // ----- Return
        return $p_list;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : properties()
    // Description :
    //   This method gives the properties of the archive.
    //   The properties are :
    //     nb : Number of files in the archive
    //     comment : Comment associated with the archive file
    //     status : not_exist, ok
    // Parameters :
    //   None
    // Return Values :
    //   0 on failure,
    //   An array with the archive properties.
    // --------------------------------------------------------------------------------
    public function properties()
    {

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            $this->privSwapBackMagicQuotes();

            return (0);
        }

        // ----- Default properties
        $v_prop            = array();
        $v_prop['comment'] = '';
        $v_prop['nb']      = 0;
        $v_prop['status']  = 'not_exist';

        // ----- Look if file exists
        if (@is_file($this->zipname)) {
            // ----- Open the zip file
            if (($this->zip_fd = @fopen($this->zipname, 'rb')) == 0) {
                $this->privSwapBackMagicQuotes();

                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open archive \'' . $this->zipname . '\' in binary read mode');

                // ----- Return
                return 0;
            }

            // ----- Read the central directory informations
            $v_central_dir = array();
            if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
                $this->privSwapBackMagicQuotes();

                return 0;
            }

            // ----- Close the zip file
            $this->privCloseFd();

            // ----- Set the user attributes
            $v_prop['comment'] = $v_central_dir['comment'];
            $v_prop['nb']      = $v_central_dir['entries'];
            $v_prop['status']  = 'ok';
        }

        // ----- Magic quotes trick
        $this->privSwapBackMagicQuotes();

        // ----- Return
        return $v_prop;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : duplicate()
    // Description :
    //   This method creates an archive by copying the content of an other one. If
    //   the archive already exist, it is replaced by the new one without any warning.
    // Parameters :
    //   $p_archive : The filename of a valid archive, or
    //                a valid PclZip object.
    // Return Values :
    //   1 on success.
    //   0 or a negative value on error (error code).
    // --------------------------------------------------------------------------------
    public function duplicate($p_archive)
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Look if the $p_archive is a PclZip object
        if ((is_object($p_archive)) && (get_class($p_archive) == 'pclzip')) {

            // ----- Duplicate the archive
            $v_result = $this->privDuplicate($p_archive->zipname);

        // ----- Look if the $p_archive is a string (so a filename)
        } elseif (is_string($p_archive)) {

            // ----- Check that $p_archive is a valid zip file
            // TBC : Should also check the archive format
            if (!is_file($p_archive)) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_MISSING_FILE, "No file with filename '" . $p_archive . "'");
                $v_result = PCLZIP_ERR_MISSING_FILE;
            } else {
                // ----- Duplicate the archive
                $v_result = $this->privDuplicate($p_archive);
            }

        // ----- Invalid variable
        } else {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid variable type p_archive_to_add");
            $v_result = PCLZIP_ERR_INVALID_PARAMETER;
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : merge()
    // Description :
    //   This method merge the $p_archive_to_add archive at the end of the current
    //   one ($this).
    //   If the archive ($this) does not exist, the merge becomes a duplicate.
    //   If the $p_archive_to_add archive does not exist, the merge is a success.
    // Parameters :
    //   $p_archive_to_add : It can be directly the filename of a valid zip archive,
    //                       or a PclZip object archive.
    // Return Values :
    //   1 on success,
    //   0 or negative values on error (see below).
    // --------------------------------------------------------------------------------
    public function merge($p_archive_to_add)
    {
        $v_result = 1;

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Check archive
        if (!$this->privCheckFormat()) {
            return (0);
        }

        // ----- Look if the $p_archive_to_add is a PclZip object
        if ((is_object($p_archive_to_add)) && (get_class($p_archive_to_add) == 'pclzip')) {

            // ----- Merge the archive
            $v_result = $this->privMerge($p_archive_to_add);

        // ----- Look if the $p_archive_to_add is a string (so a filename)
        } elseif (is_string($p_archive_to_add)) {

            // ----- Create a temporary archive
            $v_object_archive = new PclZip($p_archive_to_add);

            // ----- Merge the archive
            $v_result = $this->privMerge($v_object_archive);

        // ----- Invalid variable
        } else {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid variable type p_archive_to_add");
            $v_result = PCLZIP_ERR_INVALID_PARAMETER;
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : errorCode()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function errorCode()
    {
        if (PCLZIP_ERROR_EXTERNAL == 1) {
            return (PclErrorCode());
        }

        return ($this->error_code);
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : errorName()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function errorName($p_with_code = false)
    {
        $v_name = array(
            PCLZIP_ERR_NO_ERROR => 'PCLZIP_ERR_NO_ERROR',
            PCLZIP_ERR_WRITE_OPEN_FAIL => 'PCLZIP_ERR_WRITE_OPEN_FAIL',
            PCLZIP_ERR_READ_OPEN_FAIL => 'PCLZIP_ERR_READ_OPEN_FAIL',
            PCLZIP_ERR_INVALID_PARAMETER => 'PCLZIP_ERR_INVALID_PARAMETER',
            PCLZIP_ERR_MISSING_FILE => 'PCLZIP_ERR_MISSING_FILE',
            PCLZIP_ERR_FILENAME_TOO_LONG => 'PCLZIP_ERR_FILENAME_TOO_LONG',
            PCLZIP_ERR_INVALID_ZIP => 'PCLZIP_ERR_INVALID_ZIP',
            PCLZIP_ERR_BAD_EXTRACTED_FILE => 'PCLZIP_ERR_BAD_EXTRACTED_FILE',
            PCLZIP_ERR_DIR_CREATE_FAIL => 'PCLZIP_ERR_DIR_CREATE_FAIL',
            PCLZIP_ERR_BAD_EXTENSION => 'PCLZIP_ERR_BAD_EXTENSION',
            PCLZIP_ERR_BAD_FORMAT => 'PCLZIP_ERR_BAD_FORMAT',
            PCLZIP_ERR_DELETE_FILE_FAIL => 'PCLZIP_ERR_DELETE_FILE_FAIL',
            PCLZIP_ERR_RENAME_FILE_FAIL => 'PCLZIP_ERR_RENAME_FILE_FAIL',
            PCLZIP_ERR_BAD_CHECKSUM => 'PCLZIP_ERR_BAD_CHECKSUM',
            PCLZIP_ERR_INVALID_ARCHIVE_ZIP => 'PCLZIP_ERR_INVALID_ARCHIVE_ZIP',
            PCLZIP_ERR_MISSING_OPTION_VALUE => 'PCLZIP_ERR_MISSING_OPTION_VALUE',
            PCLZIP_ERR_INVALID_OPTION_VALUE => 'PCLZIP_ERR_INVALID_OPTION_VALUE',
            PCLZIP_ERR_UNSUPPORTED_COMPRESSION => 'PCLZIP_ERR_UNSUPPORTED_COMPRESSION',
            PCLZIP_ERR_UNSUPPORTED_ENCRYPTION => 'PCLZIP_ERR_UNSUPPORTED_ENCRYPTION',
            PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE => 'PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE',
            PCLZIP_ERR_DIRECTORY_RESTRICTION => 'PCLZIP_ERR_DIRECTORY_RESTRICTION'
        );

        if (isset($v_name[$this->error_code])) {
            $v_value = $v_name[$this->error_code];
        } else {
            $v_value = 'NoName';
        }

        if ($p_with_code) {
            return ($v_value . ' (' . $this->error_code . ')');
        }

        return ($v_value);
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : errorInfo()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function errorInfo($p_full = false)
    {
        if (PCLZIP_ERROR_EXTERNAL == 1) {
            return (PclErrorString());
        }

        if ($p_full) {
            return ($this->errorName(true) . " : " . $this->error_string);
        }

        return ($this->error_string . " [code " . $this->error_code . "]");
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // ***** UNDER THIS LINE ARE DEFINED PRIVATE INTERNAL FUNCTIONS *****
    // *****                                                        *****
    // *****       THESES FUNCTIONS MUST NOT BE USED DIRECTLY       *****
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privCheckFormat()
    // Description :
    //   This method check that the archive exists and is a valid zip archive.
    //   Several level of check exists. (futur)
    // Parameters :
    //   $p_level : Level of check. Default 0.
    //              0 : Check the first bytes (magic codes) (default value))
    //              1 : 0 + Check the central directory (futur)
    //              2 : 1 + Check each file header (futur)
    // Return Values :
    //   true on success,
    //   false on error, the error code is set.
    // --------------------------------------------------------------------------------
    public function privCheckFormat($p_level = 0)
    {
        $v_result = true;

        // ----- Reset the file system cache
        clearstatcache();

        // ----- Reset the error handler
        $this->privErrorReset();

        // ----- Look if the file exits
        if (!is_file($this->zipname)) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_MISSING_FILE, "Missing archive file '" . $this->zipname . "'");

            return (false);
        }

        // ----- Check that the file is readeable
        if (!is_readable($this->zipname)) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, "Unable to read archive '" . $this->zipname . "'");

            return (false);
        }

        // ----- Check the magic code
        // TBC

        // ----- Check the central header
        // TBC

        // ----- Check each file header
        // TBC

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privParseOptions()
    // Description :
    //   This internal methods reads the variable list of arguments ($p_options_list,
    //   $p_size) and generate an array with the options and values ($v_result_list).
    //   $v_requested_options contains the options that can be present and those that
    //   must be present.
    //   $v_requested_options is an array, with the option value as key, and 'optional',
    //   or 'mandatory' as value.
    // Parameters :
    //   See above.
    // Return Values :
    //   1 on success.
    //   0 on failure.
    // --------------------------------------------------------------------------------
    public function privParseOptions(&$p_options_list, $p_size, &$v_result_list, $v_requested_options = false)
    {
        $v_result = 1;

        // ----- Read the options
        $i = 0;
        while ($i < $p_size) {

            // ----- Check if the option is supported
            if (!isset($v_requested_options[$p_options_list[$i]])) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid optional parameter '" . $p_options_list[$i] . "' for this method");

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Look for next option
            switch ($p_options_list[$i]) {
                // ----- Look for options that request a path value
                case PCLZIP_OPT_PATH:
                case PCLZIP_OPT_REMOVE_PATH:
                case PCLZIP_OPT_ADD_PATH:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    $v_result_list[$p_options_list[$i]] = PclZipUtilTranslateWinPath($p_options_list[$i + 1], false);
                    $i++;
                    break;

                case PCLZIP_OPT_TEMP_FILE_THRESHOLD:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        return PclZip::errorCode();
                    }

                    // ----- Check for incompatible options
                    if (isset($v_result_list[PCLZIP_OPT_TEMP_FILE_OFF])) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Option '" . PclZipUtilOptionText($p_options_list[$i]) . "' can not be used with option 'PCLZIP_OPT_TEMP_FILE_OFF'");

                        return PclZip::errorCode();
                    }

                    // ----- Check the value
                    $v_value = $p_options_list[$i + 1];
                    if ((!is_integer($v_value)) || ($v_value < 0)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Integer expected for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        return PclZip::errorCode();
                    }

                    // ----- Get the value (and convert it in bytes)
                    $v_result_list[$p_options_list[$i]] = $v_value * 1048576;
                    $i++;
                    break;

                case PCLZIP_OPT_TEMP_FILE_ON:
                    // ----- Check for incompatible options
                    if (isset($v_result_list[PCLZIP_OPT_TEMP_FILE_OFF])) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Option '" . PclZipUtilOptionText($p_options_list[$i]) . "' can not be used with option 'PCLZIP_OPT_TEMP_FILE_OFF'");

                        return PclZip::errorCode();
                    }

                    $v_result_list[$p_options_list[$i]] = true;
                    break;

                case PCLZIP_OPT_TEMP_FILE_OFF:
                    // ----- Check for incompatible options
                    if (isset($v_result_list[PCLZIP_OPT_TEMP_FILE_ON])) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Option '" . PclZipUtilOptionText($p_options_list[$i]) . "' can not be used with option 'PCLZIP_OPT_TEMP_FILE_ON'");

                        return PclZip::errorCode();
                    }
                    // ----- Check for incompatible options
                    if (isset($v_result_list[PCLZIP_OPT_TEMP_FILE_THRESHOLD])) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Option '" . PclZipUtilOptionText($p_options_list[$i]) . "' can not be used with option 'PCLZIP_OPT_TEMP_FILE_THRESHOLD'");

                        return PclZip::errorCode();
                    }

                    $v_result_list[$p_options_list[$i]] = true;
                    break;

                case PCLZIP_OPT_EXTRACT_DIR_RESTRICTION:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    if (is_string($p_options_list[$i + 1]) && ($p_options_list[$i + 1] != '')) {
                        $v_result_list[$p_options_list[$i]] = PclZipUtilTranslateWinPath($p_options_list[$i + 1], false);
                        $i++;
                    } else {
                    }
                    break;

                // ----- Look for options that request an array of string for value
                case PCLZIP_OPT_BY_NAME:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    if (is_string($p_options_list[$i + 1])) {
                        $v_result_list[$p_options_list[$i]][0] = $p_options_list[$i + 1];
                    } elseif (is_array($p_options_list[$i + 1])) {
                        $v_result_list[$p_options_list[$i]] = $p_options_list[$i + 1];
                    } else {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Wrong parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }
                    $i++;
                    break;

                // ----- Look for options that request an EREG or PREG expression
                case PCLZIP_OPT_BY_EREG:
                    $p_options_list[$i] = PCLZIP_OPT_BY_PREG;
                    // ereg() is deprecated starting with PHP 5.3. Move PCLZIP_OPT_BY_EREG
                    // to PCLZIP_OPT_BY_PREG
                case PCLZIP_OPT_BY_PREG:
                    //case PCLZIP_OPT_CRYPT :
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    if (is_string($p_options_list[$i + 1])) {
                        $v_result_list[$p_options_list[$i]] = $p_options_list[$i + 1];
                    } else {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Wrong parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }
                    $i++;
                    break;

                // ----- Look for options that takes a string
                case PCLZIP_OPT_COMMENT:
                case PCLZIP_OPT_ADD_COMMENT:
                case PCLZIP_OPT_PREPEND_COMMENT:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    if (is_string($p_options_list[$i + 1])) {
                        $v_result_list[$p_options_list[$i]] = $p_options_list[$i + 1];
                    } else {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Wrong parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }
                    $i++;
                    break;

                // ----- Look for options that request an array of index
                case PCLZIP_OPT_BY_INDEX:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    $v_work_list = array();
                    if (is_string($p_options_list[$i + 1])) {

                        // ----- Remove spaces
                        $p_options_list[$i + 1] = strtr($p_options_list[$i + 1], ' ', '');

                        // ----- Parse items
                        $v_work_list = explode(",", $p_options_list[$i + 1]);
                    } elseif (is_integer($p_options_list[$i + 1])) {
                        $v_work_list[0] = $p_options_list[$i + 1] . '-' . $p_options_list[$i + 1];
                    } elseif (is_array($p_options_list[$i + 1])) {
                        $v_work_list = $p_options_list[$i + 1];
                    } else {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Value must be integer, string or array for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Reduce the index list
                    // each index item in the list must be a couple with a start and
                    // an end value : [0,3], [5-5], [8-10], ...
                    // ----- Check the format of each item
                    $v_sort_flag  = false;
                    $v_sort_value = 0;
                    for ($j = 0; $j < sizeof($v_work_list); $j++) {
                        // ----- Explode the item
                        $v_item_list      = explode("-", $v_work_list[$j]);
                        $v_size_item_list = sizeof($v_item_list);

                        // ----- TBC : Here we might check that each item is a
                        // real integer ...

                        // ----- Look for single value
                        if ($v_size_item_list == 1) {
                            // ----- Set the option value
                            $v_result_list[$p_options_list[$i]][$j]['start'] = $v_item_list[0];
                            $v_result_list[$p_options_list[$i]][$j]['end']   = $v_item_list[0];
                        } elseif ($v_size_item_list == 2) {
                            // ----- Set the option value
                            $v_result_list[$p_options_list[$i]][$j]['start'] = $v_item_list[0];
                            $v_result_list[$p_options_list[$i]][$j]['end']   = $v_item_list[1];
                        } else {
                            // ----- Error log
                            PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Too many values in index range for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                            // ----- Return
                            return PclZip::errorCode();
                        }

                        // ----- Look for list sort
                        if ($v_result_list[$p_options_list[$i]][$j]['start'] < $v_sort_value) {
                            $v_sort_flag = true;

                            // ----- TBC : An automatic sort should be writen ...
                            // ----- Error log
                            PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Invalid order of index range for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                            // ----- Return
                            return PclZip::errorCode();
                        }
                        $v_sort_value = $v_result_list[$p_options_list[$i]][$j]['start'];
                    }

                    // ----- Sort the items
                    if ($v_sort_flag) {
                        // TBC : To Be Completed
                    }

                    // ----- Next option
                    $i++;
                    break;

                // ----- Look for options that request no value
                case PCLZIP_OPT_REMOVE_ALL_PATH:
                case PCLZIP_OPT_EXTRACT_AS_STRING:
                case PCLZIP_OPT_NO_COMPRESSION:
                case PCLZIP_OPT_EXTRACT_IN_OUTPUT:
                case PCLZIP_OPT_REPLACE_NEWER:
                case PCLZIP_OPT_STOP_ON_ERROR:
                    $v_result_list[$p_options_list[$i]] = true;
                    break;

                // ----- Look for options that request an octal value
                case PCLZIP_OPT_SET_CHMOD:
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    $v_result_list[$p_options_list[$i]] = $p_options_list[$i + 1];
                    $i++;
                    break;

                // ----- Look for options that request a call-back
                case PCLZIP_CB_PRE_EXTRACT:
                case PCLZIP_CB_POST_EXTRACT:
                case PCLZIP_CB_PRE_ADD:
                case PCLZIP_CB_POST_ADD:
                    /* for futur use
                    case PCLZIP_CB_PRE_DELETE :
                    case PCLZIP_CB_POST_DELETE :
                    case PCLZIP_CB_PRE_LIST :
                    case PCLZIP_CB_POST_LIST :
                    */
                    // ----- Check the number of parameters
                    if (($i + 1) >= $p_size) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_MISSING_OPTION_VALUE, "Missing parameter value for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Get the value
                    $v_function_name = $p_options_list[$i + 1];

                    // ----- Check that the value is a valid existing function
                    if (!function_exists($v_function_name)) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_OPTION_VALUE, "Function '" . $v_function_name . "()' is not an existing function for option '" . PclZipUtilOptionText($p_options_list[$i]) . "'");

                        // ----- Return
                        return PclZip::errorCode();
                    }

                    // ----- Set the attribute
                    $v_result_list[$p_options_list[$i]] = $v_function_name;
                    $i++;
                    break;

                default:
                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Unknown parameter '" . $p_options_list[$i] . "'");

                    // ----- Return
                    return PclZip::errorCode();
            }

            // ----- Next options
            $i++;
        }

        // ----- Look for mandatory options
        if ($v_requested_options !== false) {
            for ($key = reset($v_requested_options); $key = key($v_requested_options); $key = next($v_requested_options)) {
                // ----- Look for mandatory option
                if ($v_requested_options[$key] == 'mandatory') {
                    // ----- Look if present
                    if (!isset($v_result_list[$key])) {
                        // ----- Error log
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Missing mandatory parameter " . PclZipUtilOptionText($key) . "(" . $key . ")");

                        // ----- Return
                        return PclZip::errorCode();
                    }
                }
            }
        }

        // ----- Look for default values
        if (!isset($v_result_list[PCLZIP_OPT_TEMP_FILE_THRESHOLD])) {

        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privOptionDefaultThreshold()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privOptionDefaultThreshold(&$p_options)
    {
        $v_result = 1;

        if (isset($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD]) || isset($p_options[PCLZIP_OPT_TEMP_FILE_OFF])) {
            return $v_result;
        }

        // ----- Get 'memory_limit' configuration value
        $v_memory_limit = ini_get('memory_limit');
        $v_memory_limit = trim($v_memory_limit);
        $last           = strtolower(substr($v_memory_limit, -1));
        $v_memory_limit = preg_replace('/[^0-9,.]/', '', $v_memory_limit);

        if ($last == 'g') {
            //$v_memory_limit = $v_memory_limit*1024*1024*1024;
            $v_memory_limit = $v_memory_limit * 1073741824;
        }
        if ($last == 'm') {
            //$v_memory_limit = $v_memory_limit*1024*1024;
            $v_memory_limit = $v_memory_limit * 1048576;
        }
        if ($last == 'k') {
            $v_memory_limit = $v_memory_limit * 1024;
        }

        $p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD] = floor($v_memory_limit * PCLZIP_TEMPORARY_FILE_RATIO);

        // ----- Sanity check : No threshold if value lower than 1M
        if ($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD] < 1048576) {
            unset($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD]);
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privFileDescrParseAtt()
    // Description :
    // Parameters :
    // Return Values :
    //   1 on success.
    //   0 on failure.
    // --------------------------------------------------------------------------------
    public function privFileDescrParseAtt(&$p_file_list, &$p_filedescr, $v_options, $v_requested_options = false)
    {
        $v_result = 1;

        // ----- For each file in the list check the attributes
        foreach ($p_file_list as $v_key => $v_value) {

            // ----- Check if the option is supported
            if (!isset($v_requested_options[$v_key])) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid file attribute '" . $v_key . "' for this file");

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Look for attribute
            switch ($v_key) {
                case PCLZIP_ATT_FILE_NAME:
                    if (!is_string($v_value)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid type " . gettype($v_value) . ". String expected for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    $p_filedescr['filename'] = PclZipUtilPathReduction($v_value);

                    if ($p_filedescr['filename'] == '') {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid empty filename for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    break;

                case PCLZIP_ATT_FILE_NEW_SHORT_NAME:
                    if (!is_string($v_value)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid type " . gettype($v_value) . ". String expected for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    $p_filedescr['new_short_name'] = PclZipUtilPathReduction($v_value);

                    if ($p_filedescr['new_short_name'] == '') {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid empty short filename for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }
                    break;

                case PCLZIP_ATT_FILE_NEW_FULL_NAME:
                    if (!is_string($v_value)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid type " . gettype($v_value) . ". String expected for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    $p_filedescr['new_full_name'] = PclZipUtilPathReduction($v_value);

                    if ($p_filedescr['new_full_name'] == '') {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid empty full filename for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }
                    break;

                // ----- Look for options that takes a string
                case PCLZIP_ATT_FILE_COMMENT:
                    if (!is_string($v_value)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid type " . gettype($v_value) . ". String expected for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    $p_filedescr['comment'] = $v_value;
                    break;

                case PCLZIP_ATT_FILE_MTIME:
                    if (!is_integer($v_value)) {
                        PclZip::privErrorLog(PCLZIP_ERR_INVALID_ATTRIBUTE_VALUE, "Invalid type " . gettype($v_value) . ". Integer expected for attribute '" . PclZipUtilOptionText($v_key) . "'");

                        return PclZip::errorCode();
                    }

                    $p_filedescr['mtime'] = $v_value;
                    break;

                case PCLZIP_ATT_FILE_CONTENT:
                    $p_filedescr['content'] = $v_value;
                    break;

                default:
                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Unknown parameter '" . $v_key . "'");

                    // ----- Return
                    return PclZip::errorCode();
            }

            // ----- Look for mandatory options
            if ($v_requested_options !== false) {
                for ($key = reset($v_requested_options); $key = key($v_requested_options); $key = next($v_requested_options)) {
                    // ----- Look for mandatory option
                    if ($v_requested_options[$key] == 'mandatory') {
                        // ----- Look if present
                        if (!isset($p_file_list[$key])) {
                            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Missing mandatory parameter " . PclZipUtilOptionText($key) . "(" . $key . ")");

                            return PclZip::errorCode();
                        }
                    }
                }
            }

            // end foreach
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privFileDescrExpand()
    // Description :
    //   This method look for each item of the list to see if its a file, a folder
    //   or a string to be added as file. For any other type of files (link, other)
    //   just ignore the item.
    //   Then prepare the information that will be stored for that file.
    //   When its a folder, expand the folder with all the files that are in that
    //   folder (recursively).
    // Parameters :
    // Return Values :
    //   1 on success.
    //   0 on failure.
    // --------------------------------------------------------------------------------
    public function privFileDescrExpand(&$p_filedescr_list, &$p_options)
    {
        $v_result = 1;

        // ----- Create a result list
        $v_result_list = array();

        // ----- Look each entry
        for ($i = 0; $i < sizeof($p_filedescr_list); $i++) {

            // ----- Get filedescr
            $v_descr = $p_filedescr_list[$i];

            // ----- Reduce the filename
            $v_descr['filename'] = PclZipUtilTranslateWinPath($v_descr['filename'], false);
            $v_descr['filename'] = PclZipUtilPathReduction($v_descr['filename']);

            // ----- Look for real file or folder
            if (file_exists($v_descr['filename'])) {
                if (@is_file($v_descr['filename'])) {
                    $v_descr['type'] = 'file';
                } elseif (@is_dir($v_descr['filename'])) {
                    $v_descr['type'] = 'folder';
                } elseif (@is_link($v_descr['filename'])) {
                    // skip
                    continue;
                } else {
                    // skip
                    continue;
                }

            // ----- Look for string added as file
            } elseif (isset($v_descr['content'])) {
                $v_descr['type'] = 'virtual_file';

            // ----- Missing file
            } else {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_MISSING_FILE, "File '" . $v_descr['filename'] . "' does not exist");

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Calculate the stored filename
            $this->privCalculateStoredFilename($v_descr, $p_options);

            // ----- Add the descriptor in result list
            $v_result_list[sizeof($v_result_list)] = $v_descr;

            // ----- Look for folder
            if ($v_descr['type'] == 'folder') {
                // ----- List of items in folder
                $v_dirlist_descr = array();
                $v_dirlist_nb    = 0;
                if ($v_folder_handler = @opendir($v_descr['filename'])) {
                    while (($v_item_handler = @readdir($v_folder_handler)) !== false) {

                        // ----- Skip '.' and '..'
                        if (($v_item_handler == '.') || ($v_item_handler == '..')) {
                            continue;
                        }

                        // ----- Compose the full filename
                        $v_dirlist_descr[$v_dirlist_nb]['filename'] = $v_descr['filename'] . '/' . $v_item_handler;

                        // ----- Look for different stored filename
                        // Because the name of the folder was changed, the name of the
                        // files/sub-folders also change
                        if (($v_descr['stored_filename'] != $v_descr['filename']) && (!isset($p_options[PCLZIP_OPT_REMOVE_ALL_PATH]))) {
                            if ($v_descr['stored_filename'] != '') {
                                $v_dirlist_descr[$v_dirlist_nb]['new_full_name'] = $v_descr['stored_filename'] . '/' . $v_item_handler;
                            } else {
                                $v_dirlist_descr[$v_dirlist_nb]['new_full_name'] = $v_item_handler;
                            }
                        }

                        $v_dirlist_nb++;
                    }

                    @closedir($v_folder_handler);
                } else {
                    // TBC : unable to open folder in read mode
                }

                // ----- Expand each element of the list
                if ($v_dirlist_nb != 0) {
                    // ----- Expand
                    if (($v_result = $this->privFileDescrExpand($v_dirlist_descr, $p_options)) != 1) {
                        return $v_result;
                    }

                    // ----- Concat the resulting list
                    $v_result_list = array_merge($v_result_list, $v_dirlist_descr);
                } else {
                }

                // ----- Free local array
                unset($v_dirlist_descr);
            }
        }

        // ----- Get the result list
        $p_filedescr_list = $v_result_list;

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privCreate()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privCreate($p_filedescr_list, &$p_result_list, &$p_options)
    {
        $v_result      = 1;
        $v_list_detail = array();

        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Open the file in write mode
        if (($v_result = $this->privOpenFd('wb')) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Add the list of files
        $v_result = $this->privAddList($p_filedescr_list, $p_result_list, $p_options);

        // ----- Close
        $this->privCloseFd();

        // ----- Magic quotes trick
        $this->privSwapBackMagicQuotes();

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privAdd()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privAdd($p_filedescr_list, &$p_result_list, &$p_options)
    {
        $v_result      = 1;
        $v_list_detail = array();

        // ----- Look if the archive exists or is empty
        if ((!is_file($this->zipname)) || (filesize($this->zipname) == 0)) {

            // ----- Do a create
            $v_result = $this->privCreate($p_filedescr_list, $p_result_list, $p_options);

            // ----- Return
            return $v_result;
        }
        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Open the zip file
        if (($v_result = $this->privOpenFd('rb')) != 1) {
            // ----- Magic quotes trick
            $this->privSwapBackMagicQuotes();

            // ----- Return
            return $v_result;
        }

        // ----- Read the central directory informations
        $v_central_dir = array();
        if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
            $this->privCloseFd();
            $this->privSwapBackMagicQuotes();

            return $v_result;
        }

        // ----- Go to beginning of File
        @rewind($this->zip_fd);

        // ----- Creates a temporay file
        $v_zip_temp_name = PCLZIP_TEMPORARY_DIR . uniqid('pclzip-') . '.tmp';

        // ----- Open the temporary file in write mode
        if (($v_zip_temp_fd = @fopen($v_zip_temp_name, 'wb')) == 0) {
            $this->privCloseFd();
            $this->privSwapBackMagicQuotes();

            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open temporary file \'' . $v_zip_temp_name . '\' in binary write mode');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Copy the files from the archive to the temporary file
        // TBC : Here I should better append the file and go back to erase the central dir
        $v_size = $v_central_dir['offset'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = fread($this->zip_fd, $v_read_size);
            @fwrite($v_zip_temp_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Swap the file descriptor
        // Here is a trick : I swap the temporary fd with the zip fd, in order to use
        // the following methods on the temporary fil and not the real archive
        $v_swap        = $this->zip_fd;
        $this->zip_fd  = $v_zip_temp_fd;
        $v_zip_temp_fd = $v_swap;

        // ----- Add the files
        $v_header_list = array();
        if (($v_result = $this->privAddFileList($p_filedescr_list, $v_header_list, $p_options)) != 1) {
            fclose($v_zip_temp_fd);
            $this->privCloseFd();
            @unlink($v_zip_temp_name);
            $this->privSwapBackMagicQuotes();

            // ----- Return
            return $v_result;
        }

        // ----- Store the offset of the central dir
        $v_offset = @ftell($this->zip_fd);

        // ----- Copy the block of file headers from the old archive
        $v_size = $v_central_dir['size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($v_zip_temp_fd, $v_read_size);
            @fwrite($this->zip_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Create the Central Dir files header
        for ($i = 0, $v_count = 0; $i < sizeof($v_header_list); $i++) {
            // ----- Create the file header
            if ($v_header_list[$i]['status'] == 'ok') {
                if (($v_result = $this->privWriteCentralFileHeader($v_header_list[$i])) != 1) {
                    fclose($v_zip_temp_fd);
                    $this->privCloseFd();
                    @unlink($v_zip_temp_name);
                    $this->privSwapBackMagicQuotes();

                    // ----- Return
                    return $v_result;
                }
                $v_count++;
            }

            // ----- Transform the header to a 'usable' info
            $this->privConvertHeader2FileInfo($v_header_list[$i], $p_result_list[$i]);
        }

        // ----- Zip file comment
        $v_comment = $v_central_dir['comment'];
        if (isset($p_options[PCLZIP_OPT_COMMENT])) {
            $v_comment = $p_options[PCLZIP_OPT_COMMENT];
        }
        if (isset($p_options[PCLZIP_OPT_ADD_COMMENT])) {
            $v_comment = $v_comment . $p_options[PCLZIP_OPT_ADD_COMMENT];
        }
        if (isset($p_options[PCLZIP_OPT_PREPEND_COMMENT])) {
            $v_comment = $p_options[PCLZIP_OPT_PREPEND_COMMENT] . $v_comment;
        }

        // ----- Calculate the size of the central header
        $v_size = @ftell($this->zip_fd) - $v_offset;

        // ----- Create the central dir footer
        if (($v_result = $this->privWriteCentralHeader($v_count + $v_central_dir['entries'], $v_size, $v_offset, $v_comment)) != 1) {
            // ----- Reset the file list
            unset($v_header_list);
            $this->privSwapBackMagicQuotes();

            // ----- Return
            return $v_result;
        }

        // ----- Swap back the file descriptor
        $v_swap        = $this->zip_fd;
        $this->zip_fd  = $v_zip_temp_fd;
        $v_zip_temp_fd = $v_swap;

        // ----- Close
        $this->privCloseFd();

        // ----- Close the temporary file
        @fclose($v_zip_temp_fd);

        // ----- Magic quotes trick
        $this->privSwapBackMagicQuotes();

        // ----- Delete the zip file
        // TBC : I should test the result ...
        @unlink($this->zipname);

        // ----- Rename the temporary file
        // TBC : I should test the result ...
        //@rename($v_zip_temp_name, $this->zipname);
        PclZipUtilRename($v_zip_temp_name, $this->zipname);

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privOpenFd()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function privOpenFd($p_mode)
    {
        $v_result = 1;

        // ----- Look if already open
        if ($this->zip_fd != 0) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Zip file \'' . $this->zipname . '\' already open');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Open the zip file
        if (($this->zip_fd = @fopen($this->zipname, $p_mode)) == 0) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open archive \'' . $this->zipname . '\' in ' . $p_mode . ' mode');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privCloseFd()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function privCloseFd()
    {
        $v_result = 1;

        if ($this->zip_fd != 0) {
            @fclose($this->zip_fd);
        }
        $this->zip_fd = 0;

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privAddList()
    // Description :
    //   $p_add_dir and $p_remove_dir will give the ability to memorize a path which is
    //   different from the real path of the file. This is usefull if you want to have PclTar
    //   running in any directory, and memorize relative path from an other directory.
    // Parameters :
    //   $p_list : An array containing the file or directory names to add in the tar
    //   $p_result_list : list of added files with their properties (specially the status field)
    //   $p_add_dir : Path to add in the filename path archived
    //   $p_remove_dir : Path to remove in the filename path archived
    // Return Values :
    // --------------------------------------------------------------------------------
    //  function privAddList($p_list, &$p_result_list, $p_add_dir, $p_remove_dir, $p_remove_all_dir, &$p_options)
    public function privAddList($p_filedescr_list, &$p_result_list, &$p_options)
    {
        $v_result = 1;

        // ----- Add the files
        $v_header_list = array();
        if (($v_result = $this->privAddFileList($p_filedescr_list, $v_header_list, $p_options)) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Store the offset of the central dir
        $v_offset = @ftell($this->zip_fd);

        // ----- Create the Central Dir files header
        for ($i = 0, $v_count = 0; $i < sizeof($v_header_list); $i++) {
            // ----- Create the file header
            if ($v_header_list[$i]['status'] == 'ok') {
                if (($v_result = $this->privWriteCentralFileHeader($v_header_list[$i])) != 1) {
                    // ----- Return
                    return $v_result;
                }
                $v_count++;
            }

            // ----- Transform the header to a 'usable' info
            $this->privConvertHeader2FileInfo($v_header_list[$i], $p_result_list[$i]);
        }

        // ----- Zip file comment
        $v_comment = '';
        if (isset($p_options[PCLZIP_OPT_COMMENT])) {
            $v_comment = $p_options[PCLZIP_OPT_COMMENT];
        }

        // ----- Calculate the size of the central header
        $v_size = @ftell($this->zip_fd) - $v_offset;

        // ----- Create the central dir footer
        if (($v_result = $this->privWriteCentralHeader($v_count, $v_size, $v_offset, $v_comment)) != 1) {
            // ----- Reset the file list
            unset($v_header_list);

            // ----- Return
            return $v_result;
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privAddFileList()
    // Description :
    // Parameters :
    //   $p_filedescr_list : An array containing the file description
    //                      or directory names to add in the zip
    //   $p_result_list : list of added files with their properties (specially the status field)
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privAddFileList($p_filedescr_list, &$p_result_list, &$p_options)
    {
        $v_result = 1;
        $v_header = array();

        // ----- Recuperate the current number of elt in list
        $v_nb = sizeof($p_result_list);

        // ----- Loop on the files
        for ($j = 0; ($j < sizeof($p_filedescr_list)) && ($v_result == 1); $j++) {
            // ----- Format the filename
            $p_filedescr_list[$j]['filename'] = PclZipUtilTranslateWinPath($p_filedescr_list[$j]['filename'], false);

            // ----- Skip empty file names
            // TBC : Can this be possible ? not checked in DescrParseAtt ?
            if ($p_filedescr_list[$j]['filename'] == "") {
                continue;
            }

            // ----- Check the filename
            if (($p_filedescr_list[$j]['type'] != 'virtual_file') && (!file_exists($p_filedescr_list[$j]['filename']))) {
                PclZip::privErrorLog(PCLZIP_ERR_MISSING_FILE, "File '" . $p_filedescr_list[$j]['filename'] . "' does not exist");

                return PclZip::errorCode();
            }

            // ----- Look if it is a file or a dir with no all path remove option
            // or a dir with all its path removed
            //      if (   (is_file($p_filedescr_list[$j]['filename']))
            //          || (   is_dir($p_filedescr_list[$j]['filename'])
            if (($p_filedescr_list[$j]['type'] == 'file') || ($p_filedescr_list[$j]['type'] == 'virtual_file') || (($p_filedescr_list[$j]['type'] == 'folder') && (!isset($p_options[PCLZIP_OPT_REMOVE_ALL_PATH]) || !$p_options[PCLZIP_OPT_REMOVE_ALL_PATH]))) {

                // ----- Add the file
                $v_result = $this->privAddFile($p_filedescr_list[$j], $v_header, $p_options);
                if ($v_result != 1) {
                    return $v_result;
                }

                // ----- Store the file infos
                $p_result_list[$v_nb++] = $v_header;
            }
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privAddFile()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privAddFile($p_filedescr, &$p_header, &$p_options)
    {
        $v_result = 1;

        // ----- Working variable
        $p_filename = $p_filedescr['filename'];

        // TBC : Already done in the fileAtt check ... ?
        if ($p_filename == "") {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_PARAMETER, "Invalid file list parameter (invalid or empty list)");

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Look for a stored different filename
        /* TBC : Removed
        if (isset($p_filedescr['stored_filename'])) {
        $v_stored_filename = $p_filedescr['stored_filename'];
        } else {
        $v_stored_filename = $p_filedescr['stored_filename'];
        }
        */

        // ----- Set the file properties
        clearstatcache();
        $p_header['version']           = 20;
        $p_header['version_extracted'] = 10;
        $p_header['flag']              = 0;
        $p_header['compression']       = 0;
        $p_header['crc']               = 0;
        $p_header['compressed_size']   = 0;
        $p_header['filename_len']      = strlen($p_filename);
        $p_header['extra_len']         = 0;
        $p_header['disk']              = 0;
        $p_header['internal']          = 0;
        $p_header['offset']            = 0;
        $p_header['filename']          = $p_filename;
        // TBC : Removed    $p_header['stored_filename'] = $v_stored_filename;
        $p_header['stored_filename']   = $p_filedescr['stored_filename'];
        $p_header['extra']             = '';
        $p_header['status']            = 'ok';
        $p_header['index']             = -1;

        // ----- Look for regular file
        if ($p_filedescr['type'] == 'file') {
            $p_header['external'] = 0x00000000;
            $p_header['size']     = filesize($p_filename);

        // ----- Look for regular folder
        } elseif ($p_filedescr['type'] == 'folder') {
            $p_header['external'] = 0x00000010;
            $p_header['mtime']    = filemtime($p_filename);
            $p_header['size']     = filesize($p_filename);

        // ----- Look for virtual file
        } elseif ($p_filedescr['type'] == 'virtual_file') {
            $p_header['external'] = 0x00000000;
            $p_header['size']     = strlen($p_filedescr['content']);
        }

        // ----- Look for filetime
        if (isset($p_filedescr['mtime'])) {
            $p_header['mtime'] = $p_filedescr['mtime'];
        } elseif ($p_filedescr['type'] == 'virtual_file') {
            $p_header['mtime'] = time();
        } else {
            $p_header['mtime'] = filemtime($p_filename);
        }

        // ------ Look for file comment
        if (isset($p_filedescr['comment'])) {
            $p_header['comment_len'] = strlen($p_filedescr['comment']);
            $p_header['comment']     = $p_filedescr['comment'];
        } else {
            $p_header['comment_len'] = 0;
            $p_header['comment']     = '';
        }

        // ----- Look for pre-add callback
        if (isset($p_options[PCLZIP_CB_PRE_ADD])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_header, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_PRE_ADD].'(PCLZIP_CB_PRE_ADD, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_PRE_ADD](PCLZIP_CB_PRE_ADD, $v_local_header);
            if ($v_result == 0) {
                // ----- Change the file status
                $p_header['status'] = "skipped";
                $v_result           = 1;
            }

            // ----- Update the informations
            // Only some fields can be modified
            if ($p_header['stored_filename'] != $v_local_header['stored_filename']) {
                $p_header['stored_filename'] = PclZipUtilPathReduction($v_local_header['stored_filename']);
            }
        }

        // ----- Look for empty stored filename
        if ($p_header['stored_filename'] == "") {
            $p_header['status'] = "filtered";
        }

        // ----- Check the path length
        if (strlen($p_header['stored_filename']) > 0xFF) {
            $p_header['status'] = 'filename_too_long';
        }

        // ----- Look if no error, or file not skipped
        if ($p_header['status'] == 'ok') {

            // ----- Look for a file
            if ($p_filedescr['type'] == 'file') {
                // ----- Look for using temporary file to zip
                if ((!isset($p_options[PCLZIP_OPT_TEMP_FILE_OFF])) && (isset($p_options[PCLZIP_OPT_TEMP_FILE_ON]) || (isset($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD]) && ($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD] <= $p_header['size'])))) {
                    $v_result = $this->privAddFileUsingTempFile($p_filedescr, $p_header, $p_options);
                    if ($v_result < PCLZIP_ERR_NO_ERROR) {
                        return $v_result;
                    }

                // ----- Use "in memory" zip algo
                } else {

                    // ----- Open the source file
                    if (($v_file = @fopen($p_filename, "rb")) == 0) {
                        PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, "Unable to open file '$p_filename' in binary read mode");

                        return PclZip::errorCode();
                    }

                    // ----- Read the file content
                    $v_content = @fread($v_file, $p_header['size']);

                    // ----- Close the file
                    @fclose($v_file);

                    // ----- Calculate the CRC
                    $p_header['crc'] = @crc32($v_content);

                    // ----- Look for no compression
                    if ($p_options[PCLZIP_OPT_NO_COMPRESSION]) {
                        // ----- Set header parameters
                        $p_header['compressed_size'] = $p_header['size'];
                        $p_header['compression']     = 0;

                    // ----- Look for normal compression
                    } else {
                        // ----- Compress the content
                        $v_content = @gzdeflate($v_content);

                        // ----- Set header parameters
                        $p_header['compressed_size'] = strlen($v_content);
                        $p_header['compression']     = 8;
                    }

                    // ----- Call the header generation
                    if (($v_result = $this->privWriteFileHeader($p_header)) != 1) {
                        @fclose($v_file);

                        return $v_result;
                    }

                    // ----- Write the compressed (or not) content
                    @fwrite($this->zip_fd, $v_content, $p_header['compressed_size']);

                }

            // ----- Look for a virtual file (a file from string)
            } elseif ($p_filedescr['type'] == 'virtual_file') {

                $v_content = $p_filedescr['content'];

                // ----- Calculate the CRC
                $p_header['crc'] = @crc32($v_content);

                // ----- Look for no compression
                if ($p_options[PCLZIP_OPT_NO_COMPRESSION]) {
                    // ----- Set header parameters
                    $p_header['compressed_size'] = $p_header['size'];
                    $p_header['compression']     = 0;

                // ----- Look for normal compression
                } else {
                    // ----- Compress the content
                    $v_content = @gzdeflate($v_content);

                    // ----- Set header parameters
                    $p_header['compressed_size'] = strlen($v_content);
                    $p_header['compression']     = 8;
                }

                // ----- Call the header generation
                if (($v_result = $this->privWriteFileHeader($p_header)) != 1) {
                    @fclose($v_file);

                    return $v_result;
                }

                // ----- Write the compressed (or not) content
                @fwrite($this->zip_fd, $v_content, $p_header['compressed_size']);

            // ----- Look for a directory
            } elseif ($p_filedescr['type'] == 'folder') {
                // ----- Look for directory last '/'
                if (@substr($p_header['stored_filename'], -1) != '/') {
                    $p_header['stored_filename'] .= '/';
                }

                // ----- Set the file properties
                $p_header['size']     = 0;
                //$p_header['external'] = 0x41FF0010;   // Value for a folder : to be checked
                $p_header['external'] = 0x00000010; // Value for a folder : to be checked

                // ----- Call the header generation
                if (($v_result = $this->privWriteFileHeader($p_header)) != 1) {
                    return $v_result;
                }
            }
        }

        // ----- Look for post-add callback
        if (isset($p_options[PCLZIP_CB_POST_ADD])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_header, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_POST_ADD].'(PCLZIP_CB_POST_ADD, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_POST_ADD](PCLZIP_CB_POST_ADD, $v_local_header);
            if ($v_result == 0) {
                // ----- Ignored
                $v_result = 1;
            }

            // ----- Update the informations
            // Nothing can be modified
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privAddFileUsingTempFile()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privAddFileUsingTempFile($p_filedescr, &$p_header, &$p_options)
    {
        $v_result = PCLZIP_ERR_NO_ERROR;

        // ----- Working variable
        $p_filename = $p_filedescr['filename'];

        // ----- Open the source file
        if (($v_file = @fopen($p_filename, "rb")) == 0) {
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, "Unable to open file '$p_filename' in binary read mode");

            return PclZip::errorCode();
        }

        // ----- Creates a compressed temporary file
        $v_gzip_temp_name = PCLZIP_TEMPORARY_DIR . uniqid('pclzip-') . '.gz';
        if (($v_file_compressed = @gzopen($v_gzip_temp_name, "wb")) == 0) {
            fclose($v_file);
            PclZip::privErrorLog(PCLZIP_ERR_WRITE_OPEN_FAIL, 'Unable to open temporary file \'' . $v_gzip_temp_name . '\' in binary write mode');

            return PclZip::errorCode();
        }

        // ----- Read the file by PCLZIP_READ_BLOCK_SIZE octets blocks
        $v_size = filesize($p_filename);
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($v_file, $v_read_size);
            //$v_binary_data = pack('a'.$v_read_size, $v_buffer);
            @gzputs($v_file_compressed, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Close the file
        @fclose($v_file);
        @gzclose($v_file_compressed);

        // ----- Check the minimum file size
        if (filesize($v_gzip_temp_name) < 18) {
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'gzip temporary file \'' . $v_gzip_temp_name . '\' has invalid filesize - should be minimum 18 bytes');

            return PclZip::errorCode();
        }

        // ----- Extract the compressed attributes
        if (($v_file_compressed = @fopen($v_gzip_temp_name, "rb")) == 0) {
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open temporary file \'' . $v_gzip_temp_name . '\' in binary read mode');

            return PclZip::errorCode();
        }

        // ----- Read the gzip file header
        $v_binary_data = @fread($v_file_compressed, 10);
        $v_data_header = unpack('a1id1/a1id2/a1cm/a1flag/Vmtime/a1xfl/a1os', $v_binary_data);

        // ----- Check some parameters
        $v_data_header['os'] = bin2hex($v_data_header['os']);

        // ----- Read the gzip file footer
        @fseek($v_file_compressed, filesize($v_gzip_temp_name) - 8);
        $v_binary_data = @fread($v_file_compressed, 8);
        $v_data_footer = unpack('Vcrc/Vcompressed_size', $v_binary_data);

        // ----- Set the attributes
        $p_header['compression']     = ord($v_data_header['cm']);
        //$p_header['mtime'] = $v_data_header['mtime'];
        $p_header['crc']             = $v_data_footer['crc'];
        $p_header['compressed_size'] = filesize($v_gzip_temp_name) - 18;

        // ----- Close the file
        @fclose($v_file_compressed);

        // ----- Call the header generation
        if (($v_result = $this->privWriteFileHeader($p_header)) != 1) {
            return $v_result;
        }

        // ----- Add the compressed data
        if (($v_file_compressed = @fopen($v_gzip_temp_name, "rb")) == 0) {
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open temporary file \'' . $v_gzip_temp_name . '\' in binary read mode');

            return PclZip::errorCode();
        }

        // ----- Read the file by PCLZIP_READ_BLOCK_SIZE octets blocks
        fseek($v_file_compressed, 10);
        $v_size = $p_header['compressed_size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($v_file_compressed, $v_read_size);
            //$v_binary_data = pack('a'.$v_read_size, $v_buffer);
            @fwrite($this->zip_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Close the file
        @fclose($v_file_compressed);

        // ----- Unlink the temporary file
        @unlink($v_gzip_temp_name);

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privCalculateStoredFilename()
    // Description :
    //   Based on file descriptor properties and global options, this method
    //   calculate the filename that will be stored in the archive.
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privCalculateStoredFilename(&$p_filedescr, &$p_options)
    {
        $v_result = 1;

        // ----- Working variables
        $p_filename = $p_filedescr['filename'];
        if (isset($p_options[PCLZIP_OPT_ADD_PATH])) {
            $p_add_dir = $p_options[PCLZIP_OPT_ADD_PATH];
        } else {
            $p_add_dir = '';
        }
        if (isset($p_options[PCLZIP_OPT_REMOVE_PATH])) {
            $p_remove_dir = $p_options[PCLZIP_OPT_REMOVE_PATH];
        } else {
            $p_remove_dir = '';
        }
        if (isset($p_options[PCLZIP_OPT_REMOVE_ALL_PATH])) {
            $p_remove_all_dir = $p_options[PCLZIP_OPT_REMOVE_ALL_PATH];
        } else {
            $p_remove_all_dir = 0;
        }

        // ----- Look for full name change
        if (isset($p_filedescr['new_full_name'])) {
            // ----- Remove drive letter if any
            $v_stored_filename = PclZipUtilTranslateWinPath($p_filedescr['new_full_name']);

        // ----- Look for path and/or short name change
        } else {

            // ----- Look for short name change
            // Its when we cahnge just the filename but not the path
            if (isset($p_filedescr['new_short_name'])) {
                $v_path_info = pathinfo($p_filename);
                $v_dir       = '';
                if ($v_path_info['dirname'] != '') {
                    $v_dir = $v_path_info['dirname'] . '/';
                }
                $v_stored_filename = $v_dir . $p_filedescr['new_short_name'];
            } else {
                // ----- Calculate the stored filename
                $v_stored_filename = $p_filename;
            }

            // ----- Look for all path to remove
            if ($p_remove_all_dir) {
                $v_stored_filename = basename($p_filename);

            // ----- Look for partial path remove
            } elseif ($p_remove_dir != "") {
                if (substr($p_remove_dir, -1) != '/') {
                    $p_remove_dir .= "/";
                }

                if ((substr($p_filename, 0, 2) == "./") || (substr($p_remove_dir, 0, 2) == "./")) {

                    if ((substr($p_filename, 0, 2) == "./") && (substr($p_remove_dir, 0, 2) != "./")) {
                        $p_remove_dir = "./" . $p_remove_dir;
                    }
                    if ((substr($p_filename, 0, 2) != "./") && (substr($p_remove_dir, 0, 2) == "./")) {
                        $p_remove_dir = substr($p_remove_dir, 2);
                    }
                }

                $v_compare = PclZipUtilPathInclusion($p_remove_dir, $v_stored_filename);
                if ($v_compare > 0) {
                    if ($v_compare == 2) {
                        $v_stored_filename = "";
                    } else {
                        $v_stored_filename = substr($v_stored_filename, strlen($p_remove_dir));
                    }
                }
            }

            // ----- Remove drive letter if any
            $v_stored_filename = PclZipUtilTranslateWinPath($v_stored_filename);

            // ----- Look for path to add
            if ($p_add_dir != "") {
                if (substr($p_add_dir, -1) == "/") {
                    $v_stored_filename = $p_add_dir . $v_stored_filename;
                } else {
                    $v_stored_filename = $p_add_dir . "/" . $v_stored_filename;
                }
            }
        }

        // ----- Filename (reduce the path of stored name)
        $v_stored_filename              = PclZipUtilPathReduction($v_stored_filename);
        $p_filedescr['stored_filename'] = $v_stored_filename;

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privWriteFileHeader()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privWriteFileHeader(&$p_header)
    {
        $v_result = 1;

        // ----- Store the offset position of the file
        $p_header['offset'] = ftell($this->zip_fd);

        // ----- Transform UNIX mtime to DOS format mdate/mtime
        $v_date  = getdate($p_header['mtime']);
        $v_mtime = ($v_date['hours'] << 11) + ($v_date['minutes'] << 5) + $v_date['seconds'] / 2;
        $v_mdate = (($v_date['year'] - 1980) << 9) + ($v_date['mon'] << 5) + $v_date['mday'];

        // ----- Packed data
        $v_binary_data = pack("VvvvvvVVVvv", 0x04034b50, $p_header['version_extracted'], $p_header['flag'], $p_header['compression'], $v_mtime, $v_mdate, $p_header['crc'], $p_header['compressed_size'], $p_header['size'], strlen($p_header['stored_filename']), $p_header['extra_len']);

        // ----- Write the first 148 bytes of the header in the archive
        fputs($this->zip_fd, $v_binary_data, 30);

        // ----- Write the variable fields
        if (strlen($p_header['stored_filename']) != 0) {
            fputs($this->zip_fd, $p_header['stored_filename'], strlen($p_header['stored_filename']));
        }
        if ($p_header['extra_len'] != 0) {
            fputs($this->zip_fd, $p_header['extra'], $p_header['extra_len']);
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privWriteCentralFileHeader()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privWriteCentralFileHeader(&$p_header)
    {
        $v_result = 1;

        // TBC
        //for (reset($p_header); $key = key($p_header); next($p_header)) {
        //}

        // ----- Transform UNIX mtime to DOS format mdate/mtime
        $v_date  = getdate($p_header['mtime']);
        $v_mtime = ($v_date['hours'] << 11) + ($v_date['minutes'] << 5) + $v_date['seconds'] / 2;
        $v_mdate = (($v_date['year'] - 1980) << 9) + ($v_date['mon'] << 5) + $v_date['mday'];

        // ----- Packed data
        $v_binary_data = pack("VvvvvvvVVVvvvvvVV", 0x02014b50, $p_header['version'], $p_header['version_extracted'], $p_header['flag'], $p_header['compression'], $v_mtime, $v_mdate, $p_header['crc'], $p_header['compressed_size'], $p_header['size'], strlen($p_header['stored_filename']), $p_header['extra_len'], $p_header['comment_len'], $p_header['disk'], $p_header['internal'], $p_header['external'], $p_header['offset']);

        // ----- Write the 42 bytes of the header in the zip file
        fputs($this->zip_fd, $v_binary_data, 46);

        // ----- Write the variable fields
        if (strlen($p_header['stored_filename']) != 0) {
            fputs($this->zip_fd, $p_header['stored_filename'], strlen($p_header['stored_filename']));
        }
        if ($p_header['extra_len'] != 0) {
            fputs($this->zip_fd, $p_header['extra'], $p_header['extra_len']);
        }
        if ($p_header['comment_len'] != 0) {
            fputs($this->zip_fd, $p_header['comment'], $p_header['comment_len']);
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privWriteCentralHeader()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privWriteCentralHeader($p_nb_entries, $p_size, $p_offset, $p_comment)
    {
        $v_result = 1;

        // ----- Packed data
        $v_binary_data = pack("VvvvvVVv", 0x06054b50, 0, 0, $p_nb_entries, $p_nb_entries, $p_size, $p_offset, strlen($p_comment));

        // ----- Write the 22 bytes of the header in the zip file
        fputs($this->zip_fd, $v_binary_data, 22);

        // ----- Write the variable fields
        if (strlen($p_comment) != 0) {
            fputs($this->zip_fd, $p_comment, strlen($p_comment));
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privList()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privList(&$p_list)
    {
        $v_result = 1;

        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Open the zip file
        if (($this->zip_fd = @fopen($this->zipname, 'rb')) == 0) {
            // ----- Magic quotes trick
            $this->privSwapBackMagicQuotes();

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open archive \'' . $this->zipname . '\' in binary read mode');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Read the central directory informations
        $v_central_dir = array();
        if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
            $this->privSwapBackMagicQuotes();

            return $v_result;
        }

        // ----- Go to beginning of Central Dir
        @rewind($this->zip_fd);
        if (@fseek($this->zip_fd, $v_central_dir['offset'])) {
            $this->privSwapBackMagicQuotes();

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_ARCHIVE_ZIP, 'Invalid archive size');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Read each entry
        for ($i = 0; $i < $v_central_dir['entries']; $i++) {
            // ----- Read the file header
            if (($v_result = $this->privReadCentralFileHeader($v_header)) != 1) {
                $this->privSwapBackMagicQuotes();

                return $v_result;
            }
            $v_header['index'] = $i;

            // ----- Get the only interesting attributes
            $this->privConvertHeader2FileInfo($v_header, $p_list[$i]);
            unset($v_header);
        }

        // ----- Close the zip file
        $this->privCloseFd();

        // ----- Magic quotes trick
        $this->privSwapBackMagicQuotes();

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privConvertHeader2FileInfo()
    // Description :
    //   This function takes the file informations from the central directory
    //   entries and extract the interesting parameters that will be given back.
    //   The resulting file infos are set in the array $p_info
    //     $p_info['filename'] : Filename with full path. Given by user (add),
    //                           extracted in the filesystem (extract).
    //     $p_info['stored_filename'] : Stored filename in the archive.
    //     $p_info['size'] = Size of the file.
    //     $p_info['compressed_size'] = Compressed size of the file.
    //     $p_info['mtime'] = Last modification date of the file.
    //     $p_info['comment'] = Comment associated with the file.
    //     $p_info['folder'] = true/false : indicates if the entry is a folder or not.
    //     $p_info['status'] = status of the action on the file.
    //     $p_info['crc'] = CRC of the file content.
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privConvertHeader2FileInfo($p_header, &$p_info)
    {
        $v_result = 1;

        // ----- Get the interesting attributes
        $v_temp_path               = PclZipUtilPathReduction($p_header['filename']);
        $p_info['filename']        = $v_temp_path;
        $v_temp_path               = PclZipUtilPathReduction($p_header['stored_filename']);
        $p_info['stored_filename'] = $v_temp_path;
        $p_info['size']            = $p_header['size'];
        $p_info['compressed_size'] = $p_header['compressed_size'];
        $p_info['mtime']           = $p_header['mtime'];
        $p_info['comment']         = $p_header['comment'];
        $p_info['folder']          = (($p_header['external'] & 0x00000010) == 0x00000010);
        $p_info['index']           = $p_header['index'];
        $p_info['status']          = $p_header['status'];
        $p_info['crc']             = $p_header['crc'];

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privExtractByRule()
    // Description :
    //   Extract a file or directory depending of rules (by index, by name, ...)
    // Parameters :
    //   $p_file_list : An array where will be placed the properties of each
    //                  extracted file
    //   $p_path : Path to add while writing the extracted files
    //   $p_remove_path : Path to remove (from the file memorized path) while writing the
    //                    extracted files. If the path does not match the file path,
    //                    the file is extracted with its memorized path.
    //                    $p_remove_path does not apply to 'list' mode.
    //                    $p_path and $p_remove_path are commulative.
    // Return Values :
    //   1 on success,0 or less on error (see error code list)
    // --------------------------------------------------------------------------------
    public function privExtractByRule(&$p_file_list, $p_path, $p_remove_path, $p_remove_all_path, &$p_options)
    {
        $v_result = 1;

        // ----- Magic quotes trick
        $this->privDisableMagicQuotes();

        // ----- Check the path
        if (($p_path == "") || ((substr($p_path, 0, 1) != "/") && (substr($p_path, 0, 3) != "../") && (substr($p_path, 1, 2) != ":/"))) {
            $p_path = "./" . $p_path;
        }

        // ----- Reduce the path last (and duplicated) '/'
        if (($p_path != "./") && ($p_path != "/")) {
            // ----- Look for the path end '/'
            while (substr($p_path, -1) == "/") {
                $p_path = substr($p_path, 0, strlen($p_path) - 1);
            }
        }

        // ----- Look for path to remove format (should end by /)
        if (($p_remove_path != "") && (substr($p_remove_path, -1) != '/')) {
            $p_remove_path .= '/';
        }
        $p_remove_path_size = strlen($p_remove_path);

        // ----- Open the zip file
        if (($v_result = $this->privOpenFd('rb')) != 1) {
            $this->privSwapBackMagicQuotes();

            return $v_result;
        }

        // ----- Read the central directory informations
        $v_central_dir = array();
        if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
            // ----- Close the zip file
            $this->privCloseFd();
            $this->privSwapBackMagicQuotes();

            return $v_result;
        }

        // ----- Start at beginning of Central Dir
        $v_pos_entry = $v_central_dir['offset'];

        // ----- Read each entry
        $j_start = 0;
        for ($i = 0, $v_nb_extracted = 0; $i < $v_central_dir['entries']; $i++) {

            // ----- Read next Central dir entry
            @rewind($this->zip_fd);
            if (@fseek($this->zip_fd, $v_pos_entry)) {
                // ----- Close the zip file
                $this->privCloseFd();
                $this->privSwapBackMagicQuotes();

                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_INVALID_ARCHIVE_ZIP, 'Invalid archive size');

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Read the file header
            $v_header = array();
            if (($v_result = $this->privReadCentralFileHeader($v_header)) != 1) {
                // ----- Close the zip file
                $this->privCloseFd();
                $this->privSwapBackMagicQuotes();

                return $v_result;
            }

            // ----- Store the index
            $v_header['index'] = $i;

            // ----- Store the file position
            $v_pos_entry = ftell($this->zip_fd);

            // ----- Look for the specific extract rules
            $v_extract = false;

            // ----- Look for extract by name rule
            if ((isset($p_options[PCLZIP_OPT_BY_NAME])) && ($p_options[PCLZIP_OPT_BY_NAME] != 0)) {

                // ----- Look if the filename is in the list
                for ($j = 0; ($j < sizeof($p_options[PCLZIP_OPT_BY_NAME])) && (!$v_extract); $j++) {

                    // ----- Look for a directory
                    if (substr($p_options[PCLZIP_OPT_BY_NAME][$j], -1) == "/") {

                        // ----- Look if the directory is in the filename path
                        if ((strlen($v_header['stored_filename']) > strlen($p_options[PCLZIP_OPT_BY_NAME][$j])) && (substr($v_header['stored_filename'], 0, strlen($p_options[PCLZIP_OPT_BY_NAME][$j])) == $p_options[PCLZIP_OPT_BY_NAME][$j])) {
                            $v_extract = true;
                        }

                    // ----- Look for a filename
                    } elseif ($v_header['stored_filename'] == $p_options[PCLZIP_OPT_BY_NAME][$j]) {
                        $v_extract = true;
                    }
                }
            // ----- Look for extract by ereg rule
            // ereg() is deprecated with PHP 5.3
            /*
            elseif (   (isset($p_options[PCLZIP_OPT_BY_EREG]))
            && ($p_options[PCLZIP_OPT_BY_EREG] != "")) {

            if (ereg($p_options[PCLZIP_OPT_BY_EREG], $v_header['stored_filename'])) {
            $v_extract = true;
            }
            }
            */

            // ----- Look for extract by preg rule
            } elseif ((isset($p_options[PCLZIP_OPT_BY_PREG])) && ($p_options[PCLZIP_OPT_BY_PREG] != "")) {

                if (preg_match($p_options[PCLZIP_OPT_BY_PREG], $v_header['stored_filename'])) {
                    $v_extract = true;
                }

            // ----- Look for extract by index rule
            } elseif ((isset($p_options[PCLZIP_OPT_BY_INDEX])) && ($p_options[PCLZIP_OPT_BY_INDEX] != 0)) {

                // ----- Look if the index is in the list
                for ($j = $j_start; ($j < sizeof($p_options[PCLZIP_OPT_BY_INDEX])) && (!$v_extract); $j++) {

                    if (($i >= $p_options[PCLZIP_OPT_BY_INDEX][$j]['start']) && ($i <= $p_options[PCLZIP_OPT_BY_INDEX][$j]['end'])) {
                        $v_extract = true;
                    }
                    if ($i >= $p_options[PCLZIP_OPT_BY_INDEX][$j]['end']) {
                        $j_start = $j + 1;
                    }

                    if ($p_options[PCLZIP_OPT_BY_INDEX][$j]['start'] > $i) {
                        break;
                    }
                }

            // ----- Look for no rule, which means extract all the archive
            } else {
                $v_extract = true;
            }

            // ----- Check compression method
            if (($v_extract) && (($v_header['compression'] != 8) && ($v_header['compression'] != 0))) {
                $v_header['status'] = 'unsupported_compression';

                // ----- Look for PCLZIP_OPT_STOP_ON_ERROR
                if ((isset($p_options[PCLZIP_OPT_STOP_ON_ERROR])) && ($p_options[PCLZIP_OPT_STOP_ON_ERROR] === true)) {

                    $this->privSwapBackMagicQuotes();

                    PclZip::privErrorLog(PCLZIP_ERR_UNSUPPORTED_COMPRESSION, "Filename '" . $v_header['stored_filename'] . "' is " . "compressed by an unsupported compression " . "method (" . $v_header['compression'] . ") ");

                    return PclZip::errorCode();
                }
            }

            // ----- Check encrypted files
            if (($v_extract) && (($v_header['flag'] & 1) == 1)) {
                $v_header['status'] = 'unsupported_encryption';

                // ----- Look for PCLZIP_OPT_STOP_ON_ERROR
                if ((isset($p_options[PCLZIP_OPT_STOP_ON_ERROR])) && ($p_options[PCLZIP_OPT_STOP_ON_ERROR] === true)) {

                    $this->privSwapBackMagicQuotes();

                    PclZip::privErrorLog(PCLZIP_ERR_UNSUPPORTED_ENCRYPTION, "Unsupported encryption for " . " filename '" . $v_header['stored_filename'] . "'");

                    return PclZip::errorCode();
                }
            }

            // ----- Look for real extraction
            if (($v_extract) && ($v_header['status'] != 'ok')) {
                $v_result = $this->privConvertHeader2FileInfo($v_header, $p_file_list[$v_nb_extracted++]);
                if ($v_result != 1) {
                    $this->privCloseFd();
                    $this->privSwapBackMagicQuotes();

                    return $v_result;
                }

                $v_extract = false;
            }

            // ----- Look for real extraction
            if ($v_extract) {

                // ----- Go to the file position
                @rewind($this->zip_fd);
                if (@fseek($this->zip_fd, $v_header['offset'])) {
                    // ----- Close the zip file
                    $this->privCloseFd();

                    $this->privSwapBackMagicQuotes();

                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_ARCHIVE_ZIP, 'Invalid archive size');

                    // ----- Return
                    return PclZip::errorCode();
                }

                // ----- Look for extraction as string
                if ($p_options[PCLZIP_OPT_EXTRACT_AS_STRING]) {

                    $v_string = '';

                    // ----- Extracting the file
                    $v_result1 = $this->privExtractFileAsString($v_header, $v_string, $p_options);
                    if ($v_result1 < 1) {
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result1;
                    }

                    // ----- Get the only interesting attributes
                    if (($v_result = $this->privConvertHeader2FileInfo($v_header, $p_file_list[$v_nb_extracted])) != 1) {
                        // ----- Close the zip file
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result;
                    }

                    // ----- Set the file content
                    $p_file_list[$v_nb_extracted]['content'] = $v_string;

                    // ----- Next extracted file
                    $v_nb_extracted++;

                    // ----- Look for user callback abort
                    if ($v_result1 == 2) {
                        break;
                    }

                // ----- Look for extraction in standard output
                } elseif ((isset($p_options[PCLZIP_OPT_EXTRACT_IN_OUTPUT])) && ($p_options[PCLZIP_OPT_EXTRACT_IN_OUTPUT])) {
                    // ----- Extracting the file in standard output
                    $v_result1 = $this->privExtractFileInOutput($v_header, $p_options);
                    if ($v_result1 < 1) {
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result1;
                    }

                    // ----- Get the only interesting attributes
                    if (($v_result = $this->privConvertHeader2FileInfo($v_header, $p_file_list[$v_nb_extracted++])) != 1) {
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result;
                    }

                    // ----- Look for user callback abort
                    if ($v_result1 == 2) {
                        break;
                    }

                // ----- Look for normal extraction
                } else {
                    // ----- Extracting the file
                    $v_result1 = $this->privExtractFile($v_header, $p_path, $p_remove_path, $p_remove_all_path, $p_options);
                    if ($v_result1 < 1) {
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result1;
                    }

                    // ----- Get the only interesting attributes
                    if (($v_result = $this->privConvertHeader2FileInfo($v_header, $p_file_list[$v_nb_extracted++])) != 1) {
                        // ----- Close the zip file
                        $this->privCloseFd();
                        $this->privSwapBackMagicQuotes();

                        return $v_result;
                    }

                    // ----- Look for user callback abort
                    if ($v_result1 == 2) {
                        break;
                    }
                }
            }
        }

        // ----- Close the zip file
        $this->privCloseFd();
        $this->privSwapBackMagicQuotes();

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privExtractFile()
    // Description :
    // Parameters :
    // Return Values :
    //
    // 1 : ... ?
    // PCLZIP_ERR_USER_ABORTED(2) : User ask for extraction stop in callback
    // --------------------------------------------------------------------------------
    public function privExtractFile(&$p_entry, $p_path, $p_remove_path, $p_remove_all_path, &$p_options)
    {
        $v_result = 1;

        // ----- Read the file header
        if (($v_result = $this->privReadFileHeader($v_header)) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Check that the file header is coherent with $p_entry info
        if ($this->privCheckFileHeaders($v_header, $p_entry) != 1) {
            // TBC
        }

        // ----- Look for all path to remove
        if ($p_remove_all_path == true) {
            // ----- Look for folder entry that not need to be extracted
            if (($p_entry['external'] & 0x00000010) == 0x00000010) {

                $p_entry['status'] = "filtered";

                return $v_result;
            }

            // ----- Get the basename of the path
            $p_entry['filename'] = basename($p_entry['filename']);

        // ----- Look for path to remove
        } elseif ($p_remove_path != "") {
            if (PclZipUtilPathInclusion($p_remove_path, $p_entry['filename']) == 2) {

                // ----- Change the file status
                $p_entry['status'] = "filtered";

                // ----- Return
                return $v_result;
            }

            $p_remove_path_size = strlen($p_remove_path);
            if (substr($p_entry['filename'], 0, $p_remove_path_size) == $p_remove_path) {

                // ----- Remove the path
                $p_entry['filename'] = substr($p_entry['filename'], $p_remove_path_size);

            }
        }

        // ----- Add the path
        if ($p_path != '') {
            $p_entry['filename'] = $p_path . "/" . $p_entry['filename'];
        }

        // ----- Check a base_dir_restriction
        if (isset($p_options[PCLZIP_OPT_EXTRACT_DIR_RESTRICTION])) {
            $v_inclusion = PclZipUtilPathInclusion($p_options[PCLZIP_OPT_EXTRACT_DIR_RESTRICTION], $p_entry['filename']);
            if ($v_inclusion == 0) {

                PclZip::privErrorLog(PCLZIP_ERR_DIRECTORY_RESTRICTION, "Filename '" . $p_entry['filename'] . "' is " . "outside PCLZIP_OPT_EXTRACT_DIR_RESTRICTION");

                return PclZip::errorCode();
            }
        }

        // ----- Look for pre-extract callback
        if (isset($p_options[PCLZIP_CB_PRE_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_PRE_EXTRACT].'(PCLZIP_CB_PRE_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_PRE_EXTRACT](PCLZIP_CB_PRE_EXTRACT, $v_local_header);
            if ($v_result == 0) {
                // ----- Change the file status
                $p_entry['status'] = "skipped";
                $v_result          = 1;
            }

            // ----- Look for abort result
            if ($v_result == 2) {
                // ----- This status is internal and will be changed in 'skipped'
                $p_entry['status'] = "aborted";
                $v_result          = PCLZIP_ERR_USER_ABORTED;
            }

            // ----- Update the informations
            // Only some fields can be modified
            $p_entry['filename'] = $v_local_header['filename'];
        }

        // ----- Look if extraction should be done
        if ($p_entry['status'] == 'ok') {

            // ----- Look for specific actions while the file exist
            if (file_exists($p_entry['filename'])) {

                // ----- Look if file is a directory
                if (is_dir($p_entry['filename'])) {

                    // ----- Change the file status
                    $p_entry['status'] = "already_a_directory";

                    // ----- Look for PCLZIP_OPT_STOP_ON_ERROR
                    // For historical reason first PclZip implementation does not stop
                    // when this kind of error occurs.
                    if ((isset($p_options[PCLZIP_OPT_STOP_ON_ERROR])) && ($p_options[PCLZIP_OPT_STOP_ON_ERROR] === true)) {

                        PclZip::privErrorLog(PCLZIP_ERR_ALREADY_A_DIRECTORY, "Filename '" . $p_entry['filename'] . "' is " . "already used by an existing directory");

                        return PclZip::errorCode();
                    }

                // ----- Look if file is write protected
                } elseif (!is_writeable($p_entry['filename'])) {

                    // ----- Change the file status
                    $p_entry['status'] = "write_protected";

                    // ----- Look for PCLZIP_OPT_STOP_ON_ERROR
                    // For historical reason first PclZip implementation does not stop
                    // when this kind of error occurs.
                    if ((isset($p_options[PCLZIP_OPT_STOP_ON_ERROR])) && ($p_options[PCLZIP_OPT_STOP_ON_ERROR] === true)) {

                        PclZip::privErrorLog(PCLZIP_ERR_WRITE_OPEN_FAIL, "Filename '" . $p_entry['filename'] . "' exists " . "and is write protected");

                        return PclZip::errorCode();
                    }

                // ----- Look if the extracted file is older
                } elseif (filemtime($p_entry['filename']) > $p_entry['mtime']) {
                    // ----- Change the file status
                    if ((isset($p_options[PCLZIP_OPT_REPLACE_NEWER])) && ($p_options[PCLZIP_OPT_REPLACE_NEWER] === true)) {
                    } else {
                        $p_entry['status'] = "newer_exist";

                        // ----- Look for PCLZIP_OPT_STOP_ON_ERROR
                        // For historical reason first PclZip implementation does not stop
                        // when this kind of error occurs.
                        if ((isset($p_options[PCLZIP_OPT_STOP_ON_ERROR])) && ($p_options[PCLZIP_OPT_STOP_ON_ERROR] === true)) {

                            PclZip::privErrorLog(PCLZIP_ERR_WRITE_OPEN_FAIL, "Newer version of '" . $p_entry['filename'] . "' exists " . "and option PCLZIP_OPT_REPLACE_NEWER is not selected");

                            return PclZip::errorCode();
                        }
                    }
                } else {
                }

            // ----- Check the directory availability and create it if necessary
            } else {
                if ((($p_entry['external'] & 0x00000010) == 0x00000010) || (substr($p_entry['filename'], -1) == '/')) {
                    $v_dir_to_check = $p_entry['filename'];
                } elseif (!strstr($p_entry['filename'], "/")) {
                    $v_dir_to_check = "";
                } else {
                    $v_dir_to_check = dirname($p_entry['filename']);
                }

                if (($v_result = $this->privDirCheck($v_dir_to_check, (($p_entry['external'] & 0x00000010) == 0x00000010))) != 1) {

                    // ----- Change the file status
                    $p_entry['status'] = "path_creation_fail";

                    // ----- Return
                    //return $v_result;
                    $v_result = 1;
                }
            }
        }

        // ----- Look if extraction should be done
        if ($p_entry['status'] == 'ok') {

            // ----- Do the extraction (if not a folder)
            if (!(($p_entry['external'] & 0x00000010) == 0x00000010)) {
                // ----- Look for not compressed file
                if ($p_entry['compression'] == 0) {

                    // ----- Opening destination file
                    if (($v_dest_file = @fopen($p_entry['filename'], 'wb')) == 0) {

                        // ----- Change the file status
                        $p_entry['status'] = "write_error";

                        // ----- Return
                        return $v_result;
                    }

                    // ----- Read the file by PCLZIP_READ_BLOCK_SIZE octets blocks
                    $v_size = $p_entry['compressed_size'];
                    while ($v_size != 0) {
                        $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
                        $v_buffer    = @fread($this->zip_fd, $v_read_size);
                        /* Try to speed up the code
                        $v_binary_data = pack('a'.$v_read_size, $v_buffer);
                        @fwrite($v_dest_file, $v_binary_data, $v_read_size);
                        */
                        @fwrite($v_dest_file, $v_buffer, $v_read_size);
                        $v_size -= $v_read_size;
                    }

                    // ----- Closing the destination file
                    fclose($v_dest_file);

                    // ----- Change the file mtime
                    touch($p_entry['filename'], $p_entry['mtime']);

                } else {
                    // ----- TBC
                    // Need to be finished
                    if (($p_entry['flag'] & 1) == 1) {
                        PclZip::privErrorLog(PCLZIP_ERR_UNSUPPORTED_ENCRYPTION, 'File \'' . $p_entry['filename'] . '\' is encrypted. Encrypted files are not supported.');

                        return PclZip::errorCode();
                    }

                    // ----- Look for using temporary file to unzip
                    if ((!isset($p_options[PCLZIP_OPT_TEMP_FILE_OFF])) && (isset($p_options[PCLZIP_OPT_TEMP_FILE_ON]) || (isset($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD]) && ($p_options[PCLZIP_OPT_TEMP_FILE_THRESHOLD] <= $p_entry['size'])))) {
                        $v_result = $this->privExtractFileUsingTempFile($p_entry, $p_options);
                        if ($v_result < PCLZIP_ERR_NO_ERROR) {
                            return $v_result;
                        }

                    // ----- Look for extract in memory
                    } else {

                        // ----- Read the compressed file in a buffer (one shot)
                        $v_buffer = @fread($this->zip_fd, $p_entry['compressed_size']);

                        // ----- Decompress the file
                        $v_file_content = @gzinflate($v_buffer);
                        unset($v_buffer);
                        if ($v_file_content === false) {

                            // ----- Change the file status
                            // TBC
                            $p_entry['status'] = "error";

                            return $v_result;
                        }

                        // ----- Opening destination file
                        if (($v_dest_file = @fopen($p_entry['filename'], 'wb')) == 0) {

                            // ----- Change the file status
                            $p_entry['status'] = "write_error";

                            return $v_result;
                        }

                        // ----- Write the uncompressed data
                        @fwrite($v_dest_file, $v_file_content, $p_entry['size']);
                        unset($v_file_content);

                        // ----- Closing the destination file
                        @fclose($v_dest_file);

                    }

                    // ----- Change the file mtime
                    @touch($p_entry['filename'], $p_entry['mtime']);
                }

                // ----- Look for chmod option
                if (isset($p_options[PCLZIP_OPT_SET_CHMOD])) {

                    // ----- Change the mode of the file
                    @chmod($p_entry['filename'], $p_options[PCLZIP_OPT_SET_CHMOD]);
                }

            }
        }

        // ----- Change abort status
        if ($p_entry['status'] == "aborted") {
            $p_entry['status'] = "skipped";

        // ----- Look for post-extract callback
        } elseif (isset($p_options[PCLZIP_CB_POST_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_POST_EXTRACT].'(PCLZIP_CB_POST_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_POST_EXTRACT](PCLZIP_CB_POST_EXTRACT, $v_local_header);

            // ----- Look for abort result
            if ($v_result == 2) {
                $v_result = PCLZIP_ERR_USER_ABORTED;
            }
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privExtractFileUsingTempFile()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privExtractFileUsingTempFile(&$p_entry, &$p_options)
    {
        $v_result = 1;

        // ----- Creates a temporary file
        $v_gzip_temp_name = PCLZIP_TEMPORARY_DIR . uniqid('pclzip-') . '.gz';
        if (($v_dest_file = @fopen($v_gzip_temp_name, "wb")) == 0) {
            fclose($v_file);
            PclZip::privErrorLog(PCLZIP_ERR_WRITE_OPEN_FAIL, 'Unable to open temporary file \'' . $v_gzip_temp_name . '\' in binary write mode');

            return PclZip::errorCode();
        }

        // ----- Write gz file format header
        $v_binary_data = pack('va1a1Va1a1', 0x8b1f, chr($p_entry['compression']), chr(0x00), time(), chr(0x00), chr(3));
        @fwrite($v_dest_file, $v_binary_data, 10);

        // ----- Read the file by PCLZIP_READ_BLOCK_SIZE octets blocks
        $v_size = $p_entry['compressed_size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($this->zip_fd, $v_read_size);
            //$v_binary_data = pack('a'.$v_read_size, $v_buffer);
            @fwrite($v_dest_file, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Write gz file format footer
        $v_binary_data = pack('VV', $p_entry['crc'], $p_entry['size']);
        @fwrite($v_dest_file, $v_binary_data, 8);

        // ----- Close the temporary file
        @fclose($v_dest_file);

        // ----- Opening destination file
        if (($v_dest_file = @fopen($p_entry['filename'], 'wb')) == 0) {
            $p_entry['status'] = "write_error";

            return $v_result;
        }

        // ----- Open the temporary gz file
        if (($v_src_file = @gzopen($v_gzip_temp_name, 'rb')) == 0) {
            @fclose($v_dest_file);
            $p_entry['status'] = "read_error";
            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open temporary file \'' . $v_gzip_temp_name . '\' in binary read mode');

            return PclZip::errorCode();
        }

        // ----- Read the file by PCLZIP_READ_BLOCK_SIZE octets blocks
        $v_size = $p_entry['size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @gzread($v_src_file, $v_read_size);
            //$v_binary_data = pack('a'.$v_read_size, $v_buffer);
            @fwrite($v_dest_file, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }
        @fclose($v_dest_file);
        @gzclose($v_src_file);

        // ----- Delete the temporary file
        @unlink($v_gzip_temp_name);

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privExtractFileInOutput()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privExtractFileInOutput(&$p_entry, &$p_options)
    {
        $v_result = 1;

        // ----- Read the file header
        if (($v_result = $this->privReadFileHeader($v_header)) != 1) {
            return $v_result;
        }

        // ----- Check that the file header is coherent with $p_entry info
        if ($this->privCheckFileHeaders($v_header, $p_entry) != 1) {
            // TBC
        }

        // ----- Look for pre-extract callback
        if (isset($p_options[PCLZIP_CB_PRE_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_PRE_EXTRACT].'(PCLZIP_CB_PRE_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_PRE_EXTRACT](PCLZIP_CB_PRE_EXTRACT, $v_local_header);
            if ($v_result == 0) {
                // ----- Change the file status
                $p_entry['status'] = "skipped";
                $v_result          = 1;
            }

            // ----- Look for abort result
            if ($v_result == 2) {
                // ----- This status is internal and will be changed in 'skipped'
                $p_entry['status'] = "aborted";
                $v_result          = PCLZIP_ERR_USER_ABORTED;
            }

            // ----- Update the informations
            // Only some fields can be modified
            $p_entry['filename'] = $v_local_header['filename'];
        }

        // ----- Trace

        // ----- Look if extraction should be done
        if ($p_entry['status'] == 'ok') {

            // ----- Do the extraction (if not a folder)
            if (!(($p_entry['external'] & 0x00000010) == 0x00000010)) {
                // ----- Look for not compressed file
                if ($p_entry['compressed_size'] == $p_entry['size']) {

                    // ----- Read the file in a buffer (one shot)
                    $v_buffer = @fread($this->zip_fd, $p_entry['compressed_size']);

                    // ----- Send the file to the output
                    echo $v_buffer;
                    unset($v_buffer);
                } else {

                    // ----- Read the compressed file in a buffer (one shot)
                    $v_buffer = @fread($this->zip_fd, $p_entry['compressed_size']);

                    // ----- Decompress the file
                    $v_file_content = gzinflate($v_buffer);
                    unset($v_buffer);

                    // ----- Send the file to the output
                    echo $v_file_content;
                    unset($v_file_content);
                }
            }
        }

        // ----- Change abort status
        if ($p_entry['status'] == "aborted") {
            $p_entry['status'] = "skipped";

        // ----- Look for post-extract callback
        } elseif (isset($p_options[PCLZIP_CB_POST_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_POST_EXTRACT].'(PCLZIP_CB_POST_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_POST_EXTRACT](PCLZIP_CB_POST_EXTRACT, $v_local_header);

            // ----- Look for abort result
            if ($v_result == 2) {
                $v_result = PCLZIP_ERR_USER_ABORTED;
            }
        }

        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privExtractFileAsString()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privExtractFileAsString(&$p_entry, &$p_string, &$p_options)
    {
        $v_result = 1;

        // ----- Read the file header
        $v_header = array();
        if (($v_result = $this->privReadFileHeader($v_header)) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Check that the file header is coherent with $p_entry info
        if ($this->privCheckFileHeaders($v_header, $p_entry) != 1) {
            // TBC
        }

        // ----- Look for pre-extract callback
        if (isset($p_options[PCLZIP_CB_PRE_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_PRE_EXTRACT].'(PCLZIP_CB_PRE_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_PRE_EXTRACT](PCLZIP_CB_PRE_EXTRACT, $v_local_header);
            if ($v_result == 0) {
                // ----- Change the file status
                $p_entry['status'] = "skipped";
                $v_result          = 1;
            }

            // ----- Look for abort result
            if ($v_result == 2) {
                // ----- This status is internal and will be changed in 'skipped'
                $p_entry['status'] = "aborted";
                $v_result          = PCLZIP_ERR_USER_ABORTED;
            }

            // ----- Update the informations
            // Only some fields can be modified
            $p_entry['filename'] = $v_local_header['filename'];
        }

        // ----- Look if extraction should be done
        if ($p_entry['status'] == 'ok') {

            // ----- Do the extraction (if not a folder)
            if (!(($p_entry['external'] & 0x00000010) == 0x00000010)) {
                // ----- Look for not compressed file
                //      if ($p_entry['compressed_size'] == $p_entry['size'])
                if ($p_entry['compression'] == 0) {

                    // ----- Reading the file
                    $p_string = @fread($this->zip_fd, $p_entry['compressed_size']);
                } else {

                    // ----- Reading the file
                    $v_data = @fread($this->zip_fd, $p_entry['compressed_size']);

                    // ----- Decompress the file
                    if (($p_string = @gzinflate($v_data)) === false) {
                        // TBC
                    }
                }

                // ----- Trace
            } else {
                // TBC : error : can not extract a folder in a string
            }

        }

        // ----- Change abort status
        if ($p_entry['status'] == "aborted") {
            $p_entry['status'] = "skipped";

        // ----- Look for post-extract callback
        } elseif (isset($p_options[PCLZIP_CB_POST_EXTRACT])) {

            // ----- Generate a local information
            $v_local_header = array();
            $this->privConvertHeader2FileInfo($p_entry, $v_local_header);

            // ----- Swap the content to header
            $v_local_header['content'] = $p_string;
            $p_string                  = '';

            // ----- Call the callback
            // Here I do not use call_user_func() because I need to send a reference to the
            // header.
            //      eval('$v_result = '.$p_options[PCLZIP_CB_POST_EXTRACT].'(PCLZIP_CB_POST_EXTRACT, $v_local_header);');
            $v_result = $p_options[PCLZIP_CB_POST_EXTRACT](PCLZIP_CB_POST_EXTRACT, $v_local_header);

            // ----- Swap back the content to header
            $p_string = $v_local_header['content'];
            unset($v_local_header['content']);

            // ----- Look for abort result
            if ($v_result == 2) {
                $v_result = PCLZIP_ERR_USER_ABORTED;
            }
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privReadFileHeader()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privReadFileHeader(&$p_header)
    {
        $v_result = 1;

        // ----- Read the 4 bytes signature
        $v_binary_data = @fread($this->zip_fd, 4);
        $v_data        = unpack('Vid', $v_binary_data);

        // ----- Check signature
        if ($v_data['id'] != 0x04034b50) {

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'Invalid archive structure');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Read the first 42 bytes of the header
        $v_binary_data = fread($this->zip_fd, 26);

        // ----- Look for invalid block size
        if (strlen($v_binary_data) != 26) {
            $p_header['filename'] = "";
            $p_header['status']   = "invalid_header";

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, "Invalid block size : " . strlen($v_binary_data));

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Extract the values
        $v_data = unpack('vversion/vflag/vcompression/vmtime/vmdate/Vcrc/Vcompressed_size/Vsize/vfilename_len/vextra_len', $v_binary_data);

        // ----- Get filename
        $p_header['filename'] = fread($this->zip_fd, $v_data['filename_len']);

        // ----- Get extra_fields
        if ($v_data['extra_len'] != 0) {
            $p_header['extra'] = fread($this->zip_fd, $v_data['extra_len']);
        } else {
            $p_header['extra'] = '';
        }

        // ----- Extract properties
        $p_header['version_extracted'] = $v_data['version'];
        $p_header['compression']       = $v_data['compression'];
        $p_header['size']              = $v_data['size'];
        $p_header['compressed_size']   = $v_data['compressed_size'];
        $p_header['crc']               = $v_data['crc'];
        $p_header['flag']              = $v_data['flag'];
        $p_header['filename_len']      = $v_data['filename_len'];

        // ----- Recuperate date in UNIX format
        $p_header['mdate'] = $v_data['mdate'];
        $p_header['mtime'] = $v_data['mtime'];
        if ($p_header['mdate'] && $p_header['mtime']) {
            // ----- Extract time
            $v_hour    = ($p_header['mtime'] & 0xF800) >> 11;
            $v_minute  = ($p_header['mtime'] & 0x07E0) >> 5;
            $v_seconde = ($p_header['mtime'] & 0x001F) * 2;

            // ----- Extract date
            $v_year  = (($p_header['mdate'] & 0xFE00) >> 9) + 1980;
            $v_month = ($p_header['mdate'] & 0x01E0) >> 5;
            $v_day   = $p_header['mdate'] & 0x001F;

            // ----- Get UNIX date format
            $p_header['mtime'] = @mktime($v_hour, $v_minute, $v_seconde, $v_month, $v_day, $v_year);

        } else {
            $p_header['mtime'] = time();
        }

        // TBC
        //for (reset($v_data); $key = key($v_data); next($v_data)) {
        //}

        // ----- Set the stored filename
        $p_header['stored_filename'] = $p_header['filename'];

        // ----- Set the status field
        $p_header['status'] = "ok";

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privReadCentralFileHeader()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privReadCentralFileHeader(&$p_header)
    {
        $v_result = 1;

        // ----- Read the 4 bytes signature
        $v_binary_data = @fread($this->zip_fd, 4);
        $v_data        = unpack('Vid', $v_binary_data);

        // ----- Check signature
        if ($v_data['id'] != 0x02014b50) {

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'Invalid archive structure');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Read the first 42 bytes of the header
        $v_binary_data = fread($this->zip_fd, 42);

        // ----- Look for invalid block size
        if (strlen($v_binary_data) != 42) {
            $p_header['filename'] = "";
            $p_header['status']   = "invalid_header";

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, "Invalid block size : " . strlen($v_binary_data));

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Extract the values
        $p_header = unpack('vversion/vversion_extracted/vflag/vcompression/vmtime/vmdate/Vcrc/Vcompressed_size/Vsize/vfilename_len/vextra_len/vcomment_len/vdisk/vinternal/Vexternal/Voffset', $v_binary_data);

        // ----- Get filename
        if ($p_header['filename_len'] != 0) {
            $p_header['filename'] = fread($this->zip_fd, $p_header['filename_len']);
        } else {
            $p_header['filename'] = '';
        }

        // ----- Get extra
        if ($p_header['extra_len'] != 0) {
            $p_header['extra'] = fread($this->zip_fd, $p_header['extra_len']);
        } else {
            $p_header['extra'] = '';
        }

        // ----- Get comment
        if ($p_header['comment_len'] != 0) {
            $p_header['comment'] = fread($this->zip_fd, $p_header['comment_len']);
        } else {
            $p_header['comment'] = '';
        }

        // ----- Extract properties

        // ----- Recuperate date in UNIX format
        //if ($p_header['mdate'] && $p_header['mtime'])
        // TBC : bug : this was ignoring time with 0/0/0
        if (1) {
            // ----- Extract time
            $v_hour    = ($p_header['mtime'] & 0xF800) >> 11;
            $v_minute  = ($p_header['mtime'] & 0x07E0) >> 5;
            $v_seconde = ($p_header['mtime'] & 0x001F) * 2;

            // ----- Extract date
            $v_year  = (($p_header['mdate'] & 0xFE00) >> 9) + 1980;
            $v_month = ($p_header['mdate'] & 0x01E0) >> 5;
            $v_day   = $p_header['mdate'] & 0x001F;

            // ----- Get UNIX date format
            $p_header['mtime'] = @mktime($v_hour, $v_minute, $v_seconde, $v_month, $v_day, $v_year);

        } else {
            $p_header['mtime'] = time();
        }

        // ----- Set the stored filename
        $p_header['stored_filename'] = $p_header['filename'];

        // ----- Set default status to ok
        $p_header['status'] = 'ok';

        // ----- Look if it is a directory
        if (substr($p_header['filename'], -1) == '/') {
            //$p_header['external'] = 0x41FF0010;
            $p_header['external'] = 0x00000010;
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privCheckFileHeaders()
    // Description :
    // Parameters :
    // Return Values :
    //   1 on success,
    //   0 on error;
    // --------------------------------------------------------------------------------
    public function privCheckFileHeaders(&$p_local_header, &$p_central_header)
    {
        $v_result = 1;

        // ----- Check the static values
        // TBC
        if ($p_local_header['filename'] != $p_central_header['filename']) {
        }
        if ($p_local_header['version_extracted'] != $p_central_header['version_extracted']) {
        }
        if ($p_local_header['flag'] != $p_central_header['flag']) {
        }
        if ($p_local_header['compression'] != $p_central_header['compression']) {
        }
        if ($p_local_header['mtime'] != $p_central_header['mtime']) {
        }
        if ($p_local_header['filename_len'] != $p_central_header['filename_len']) {
        }

        // ----- Look for flag bit 3
        if (($p_local_header['flag'] & 8) == 8) {
            $p_local_header['size']            = $p_central_header['size'];
            $p_local_header['compressed_size'] = $p_central_header['compressed_size'];
            $p_local_header['crc']             = $p_central_header['crc'];
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privReadEndCentralDir()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privReadEndCentralDir(&$p_central_dir)
    {
        $v_result = 1;

        // ----- Go to the end of the zip file
        $v_size = filesize($this->zipname);
        @fseek($this->zip_fd, $v_size);
        if (@ftell($this->zip_fd) != $v_size) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'Unable to go to the end of the archive \'' . $this->zipname . '\'');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- First try : look if this is an archive with no commentaries (most of the time)
        // in this case the end of central dir is at 22 bytes of the file end
        $v_found = 0;
        if ($v_size > 26) {
            @fseek($this->zip_fd, $v_size - 22);
            if (($v_pos = @ftell($this->zip_fd)) != ($v_size - 22)) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'Unable to seek back to the middle of the archive \'' . $this->zipname . '\'');

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Read for bytes
            $v_binary_data = @fread($this->zip_fd, 4);
            $v_data        = @unpack('Vid', $v_binary_data);

            // ----- Check signature
            if ($v_data['id'] == 0x06054b50) {
                $v_found = 1;
            }

            $v_pos = ftell($this->zip_fd);
        }

        // ----- Go back to the maximum possible size of the Central Dir End Record
        if (!$v_found) {
            $v_maximum_size = 65557; // 0xFFFF + 22;
            if ($v_maximum_size > $v_size) {
                $v_maximum_size = $v_size;
            }
            @fseek($this->zip_fd, $v_size - $v_maximum_size);
            if (@ftell($this->zip_fd) != ($v_size - $v_maximum_size)) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'Unable to seek back to the middle of the archive \'' . $this->zipname . '\'');

                // ----- Return
                return PclZip::errorCode();
            }

            // ----- Read byte per byte in order to find the signature
            $v_pos   = ftell($this->zip_fd);
            $v_bytes = 0x00000000;
            while ($v_pos < $v_size) {
                // ----- Read a byte
                $v_byte = @fread($this->zip_fd, 1);

                // -----  Add the byte
                //$v_bytes = ($v_bytes << 8) | Ord($v_byte);
                // Note we mask the old value down such that once shifted we can never end up with more than a 32bit number
                // Otherwise on systems where we have 64bit integers the check below for the magic number will fail.
                $v_bytes = (($v_bytes & 0xFFFFFF) << 8) | ord($v_byte);

                // ----- Compare the bytes
                if ($v_bytes == 0x504b0506) {
                    $v_pos++;
                    break;
                }

                $v_pos++;
            }

            // ----- Look if not found end of central dir
            if ($v_pos == $v_size) {

                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, "Unable to find End of Central Dir Record signature");

                // ----- Return
                return PclZip::errorCode();
            }
        }

        // ----- Read the first 18 bytes of the header
        $v_binary_data = fread($this->zip_fd, 18);

        // ----- Look for invalid block size
        if (strlen($v_binary_data) != 18) {

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, "Invalid End of Central Dir Record size : " . strlen($v_binary_data));

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Extract the values
        $v_data = unpack('vdisk/vdisk_start/vdisk_entries/ventries/Vsize/Voffset/vcomment_size', $v_binary_data);

        // ----- Check the global size
        if (($v_pos + $v_data['comment_size'] + 18) != $v_size) {

            // ----- Removed in release 2.2 see readme file
            // The check of the file size is a little too strict.
            // Some bugs where found when a zip is encrypted/decrypted with 'crypt'.
            // While decrypted, zip has training 0 bytes
            if (0) {
                // ----- Error log
                PclZip::privErrorLog(PCLZIP_ERR_BAD_FORMAT, 'The central dir is not at the end of the archive.' . ' Some trailing bytes exists after the archive.');

                // ----- Return
                return PclZip::errorCode();
            }
        }

        // ----- Get comment
        if ($v_data['comment_size'] != 0) {
            $p_central_dir['comment'] = fread($this->zip_fd, $v_data['comment_size']);
        } else {
            $p_central_dir['comment'] = '';
        }

        $p_central_dir['entries']      = $v_data['entries'];
        $p_central_dir['disk_entries'] = $v_data['disk_entries'];
        $p_central_dir['offset']       = $v_data['offset'];
        $p_central_dir['size']         = $v_data['size'];
        $p_central_dir['disk']         = $v_data['disk'];
        $p_central_dir['disk_start']   = $v_data['disk_start'];

        // TBC
        //for (reset($p_central_dir); $key = key($p_central_dir); next($p_central_dir)) {
        //}

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privDeleteByRule()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privDeleteByRule(&$p_result_list, &$p_options)
    {
        $v_result      = 1;
        $v_list_detail = array();

        // ----- Open the zip file
        if (($v_result = $this->privOpenFd('rb')) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Read the central directory informations
        $v_central_dir = array();
        if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
            $this->privCloseFd();

            return $v_result;
        }

        // ----- Go to beginning of File
        @rewind($this->zip_fd);

        // ----- Scan all the files
        // ----- Start at beginning of Central Dir
        $v_pos_entry = $v_central_dir['offset'];
        @rewind($this->zip_fd);
        if (@fseek($this->zip_fd, $v_pos_entry)) {
            // ----- Close the zip file
            $this->privCloseFd();

            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_INVALID_ARCHIVE_ZIP, 'Invalid archive size');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Read each entry
        $v_header_list = array();
        $j_start       = 0;
        for ($i = 0, $v_nb_extracted = 0; $i < $v_central_dir['entries']; $i++) {

            // ----- Read the file header
            $v_header_list[$v_nb_extracted] = array();
            if (($v_result = $this->privReadCentralFileHeader($v_header_list[$v_nb_extracted])) != 1) {
                // ----- Close the zip file
                $this->privCloseFd();

                return $v_result;
            }

            // ----- Store the index
            $v_header_list[$v_nb_extracted]['index'] = $i;

            // ----- Look for the specific extract rules
            $v_found = false;

            // ----- Look for extract by name rule
            if ((isset($p_options[PCLZIP_OPT_BY_NAME])) && ($p_options[PCLZIP_OPT_BY_NAME] != 0)) {

                // ----- Look if the filename is in the list
                for ($j = 0; ($j < sizeof($p_options[PCLZIP_OPT_BY_NAME])) && (!$v_found); $j++) {

                    // ----- Look for a directory
                    if (substr($p_options[PCLZIP_OPT_BY_NAME][$j], -1) == "/") {

                        // ----- Look if the directory is in the filename path
                        if ((strlen($v_header_list[$v_nb_extracted]['stored_filename']) > strlen($p_options[PCLZIP_OPT_BY_NAME][$j])) && (substr($v_header_list[$v_nb_extracted]['stored_filename'], 0, strlen($p_options[PCLZIP_OPT_BY_NAME][$j])) == $p_options[PCLZIP_OPT_BY_NAME][$j])) {
                            $v_found = true;
                        } elseif ((($v_header_list[$v_nb_extracted]['external'] & 0x00000010) == 0x00000010) /* Indicates a folder */ && ($v_header_list[$v_nb_extracted]['stored_filename'] . '/' == $p_options[PCLZIP_OPT_BY_NAME][$j])) {
                            $v_found = true;
                        }

                    // ----- Look for a filename
                    } elseif ($v_header_list[$v_nb_extracted]['stored_filename'] == $p_options[PCLZIP_OPT_BY_NAME][$j]) {
                        $v_found = true;
                    }
                }

            // ----- Look for extract by ereg rule
            // ereg() is deprecated with PHP 5.3
            /*
            elseif (   (isset($p_options[PCLZIP_OPT_BY_EREG]))
            && ($p_options[PCLZIP_OPT_BY_EREG] != "")) {

            if (ereg($p_options[PCLZIP_OPT_BY_EREG], $v_header_list[$v_nb_extracted]['stored_filename'])) {
            $v_found = true;
            }
            }
            */

            // ----- Look for extract by preg rule
            } elseif ((isset($p_options[PCLZIP_OPT_BY_PREG])) && ($p_options[PCLZIP_OPT_BY_PREG] != "")) {

                if (preg_match($p_options[PCLZIP_OPT_BY_PREG], $v_header_list[$v_nb_extracted]['stored_filename'])) {
                    $v_found = true;
                }

            // ----- Look for extract by index rule
            } elseif ((isset($p_options[PCLZIP_OPT_BY_INDEX])) && ($p_options[PCLZIP_OPT_BY_INDEX] != 0)) {

                // ----- Look if the index is in the list
                for ($j = $j_start; ($j < sizeof($p_options[PCLZIP_OPT_BY_INDEX])) && (!$v_found); $j++) {

                    if (($i >= $p_options[PCLZIP_OPT_BY_INDEX][$j]['start']) && ($i <= $p_options[PCLZIP_OPT_BY_INDEX][$j]['end'])) {
                        $v_found = true;
                    }
                    if ($i >= $p_options[PCLZIP_OPT_BY_INDEX][$j]['end']) {
                        $j_start = $j + 1;
                    }

                    if ($p_options[PCLZIP_OPT_BY_INDEX][$j]['start'] > $i) {
                        break;
                    }
                }
            } else {
                $v_found = true;
            }

            // ----- Look for deletion
            if ($v_found) {
                unset($v_header_list[$v_nb_extracted]);
            } else {
                $v_nb_extracted++;
            }
        }

        // ----- Look if something need to be deleted
        if ($v_nb_extracted > 0) {

            // ----- Creates a temporay file
            $v_zip_temp_name = PCLZIP_TEMPORARY_DIR . uniqid('pclzip-') . '.tmp';

            // ----- Creates a temporary zip archive
            $v_temp_zip = new PclZip($v_zip_temp_name);

            // ----- Open the temporary zip file in write mode
            if (($v_result = $v_temp_zip->privOpenFd('wb')) != 1) {
                $this->privCloseFd();

                // ----- Return
                return $v_result;
            }

            // ----- Look which file need to be kept
            for ($i = 0; $i < sizeof($v_header_list); $i++) {

                // ----- Calculate the position of the header
                @rewind($this->zip_fd);
                if (@fseek($this->zip_fd, $v_header_list[$i]['offset'])) {
                    // ----- Close the zip file
                    $this->privCloseFd();
                    $v_temp_zip->privCloseFd();
                    @unlink($v_zip_temp_name);

                    // ----- Error log
                    PclZip::privErrorLog(PCLZIP_ERR_INVALID_ARCHIVE_ZIP, 'Invalid archive size');

                    // ----- Return
                    return PclZip::errorCode();
                }

                // ----- Read the file header
                $v_local_header = array();
                if (($v_result = $this->privReadFileHeader($v_local_header)) != 1) {
                    // ----- Close the zip file
                    $this->privCloseFd();
                    $v_temp_zip->privCloseFd();
                    @unlink($v_zip_temp_name);

                    // ----- Return
                    return $v_result;
                }

                // ----- Check that local file header is same as central file header
                if ($this->privCheckFileHeaders($v_local_header, $v_header_list[$i]) != 1) {
                    // TBC
                }
                unset($v_local_header);

                // ----- Write the file header
                if (($v_result = $v_temp_zip->privWriteFileHeader($v_header_list[$i])) != 1) {
                    // ----- Close the zip file
                    $this->privCloseFd();
                    $v_temp_zip->privCloseFd();
                    @unlink($v_zip_temp_name);

                    // ----- Return
                    return $v_result;
                }

                // ----- Read/write the data block
                if (($v_result = PclZipUtilCopyBlock($this->zip_fd, $v_temp_zip->zip_fd, $v_header_list[$i]['compressed_size'])) != 1) {
                    // ----- Close the zip file
                    $this->privCloseFd();
                    $v_temp_zip->privCloseFd();
                    @unlink($v_zip_temp_name);

                    // ----- Return
                    return $v_result;
                }
            }

            // ----- Store the offset of the central dir
            $v_offset = @ftell($v_temp_zip->zip_fd);

            // ----- Re-Create the Central Dir files header
            for ($i = 0; $i < sizeof($v_header_list); $i++) {
                // ----- Create the file header
                if (($v_result = $v_temp_zip->privWriteCentralFileHeader($v_header_list[$i])) != 1) {
                    $v_temp_zip->privCloseFd();
                    $this->privCloseFd();
                    @unlink($v_zip_temp_name);

                    // ----- Return
                    return $v_result;
                }

                // ----- Transform the header to a 'usable' info
                $v_temp_zip->privConvertHeader2FileInfo($v_header_list[$i], $p_result_list[$i]);
            }

            // ----- Zip file comment
            $v_comment = '';
            if (isset($p_options[PCLZIP_OPT_COMMENT])) {
                $v_comment = $p_options[PCLZIP_OPT_COMMENT];
            }

            // ----- Calculate the size of the central header
            $v_size = @ftell($v_temp_zip->zip_fd) - $v_offset;

            // ----- Create the central dir footer
            if (($v_result = $v_temp_zip->privWriteCentralHeader(sizeof($v_header_list), $v_size, $v_offset, $v_comment)) != 1) {
                // ----- Reset the file list
                unset($v_header_list);
                $v_temp_zip->privCloseFd();
                $this->privCloseFd();
                @unlink($v_zip_temp_name);

                // ----- Return
                return $v_result;
            }

            // ----- Close
            $v_temp_zip->privCloseFd();
            $this->privCloseFd();

            // ----- Delete the zip file
            // TBC : I should test the result ...
            @unlink($this->zipname);

            // ----- Rename the temporary file
            // TBC : I should test the result ...
            //@rename($v_zip_temp_name, $this->zipname);
            PclZipUtilRename($v_zip_temp_name, $this->zipname);

            // ----- Destroy the temporary archive
            unset($v_temp_zip);

        // ----- Remove every files : reset the file
        } elseif ($v_central_dir['entries'] != 0) {
            $this->privCloseFd();

            if (($v_result = $this->privOpenFd('wb')) != 1) {
                return $v_result;
            }

            if (($v_result = $this->privWriteCentralHeader(0, 0, 0, '')) != 1) {
                return $v_result;
            }

            $this->privCloseFd();
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privDirCheck()
    // Description :
    //   Check if a directory exists, if not it creates it and all the parents directory
    //   which may be useful.
    // Parameters :
    //   $p_dir : Directory path to check.
    // Return Values :
    //    1 : OK
    //   -1 : Unable to create directory
    // --------------------------------------------------------------------------------
    public function privDirCheck($p_dir, $p_is_dir = false)
    {
        $v_result = 1;

        // ----- Remove the final '/'
        if (($p_is_dir) && (substr($p_dir, -1) == '/')) {
            $p_dir = substr($p_dir, 0, strlen($p_dir) - 1);
        }

        // ----- Check the directory availability
        if ((is_dir($p_dir)) || ($p_dir == "")) {
            return 1;
        }

        // ----- Extract parent directory
        $p_parent_dir = dirname($p_dir);

        // ----- Just a check
        if ($p_parent_dir != $p_dir) {
            // ----- Look for parent directory
            if ($p_parent_dir != "") {
                if (($v_result = $this->privDirCheck($p_parent_dir)) != 1) {
                    return $v_result;
                }
            }
        }

        // ----- Create the directory
        if (!@mkdir($p_dir, 0777)) {
            // ----- Error log
            PclZip::privErrorLog(PCLZIP_ERR_DIR_CREATE_FAIL, "Unable to create directory '$p_dir'");

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privMerge()
    // Description :
    //   If $p_archive_to_add does not exist, the function exit with a success result.
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privMerge(&$p_archive_to_add)
    {
        $v_result = 1;

        // ----- Look if the archive_to_add exists
        if (!is_file($p_archive_to_add->zipname)) {

            // ----- Nothing to merge, so merge is a success
            $v_result = 1;

            // ----- Return
            return $v_result;
        }

        // ----- Look if the archive exists
        if (!is_file($this->zipname)) {

            // ----- Do a duplicate
            $v_result = $this->privDuplicate($p_archive_to_add->zipname);

            // ----- Return
            return $v_result;
        }

        // ----- Open the zip file
        if (($v_result = $this->privOpenFd('rb')) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Read the central directory informations
        $v_central_dir = array();
        if (($v_result = $this->privReadEndCentralDir($v_central_dir)) != 1) {
            $this->privCloseFd();

            return $v_result;
        }

        // ----- Go to beginning of File
        @rewind($this->zip_fd);

        // ----- Open the archive_to_add file
        if (($v_result = $p_archive_to_add->privOpenFd('rb')) != 1) {
            $this->privCloseFd();

            // ----- Return
            return $v_result;
        }

        // ----- Read the central directory informations
        $v_central_dir_to_add = array();
        if (($v_result = $p_archive_to_add->privReadEndCentralDir($v_central_dir_to_add)) != 1) {
            $this->privCloseFd();
            $p_archive_to_add->privCloseFd();

            return $v_result;
        }

        // ----- Go to beginning of File
        @rewind($p_archive_to_add->zip_fd);

        // ----- Creates a temporay file
        $v_zip_temp_name = PCLZIP_TEMPORARY_DIR . uniqid('pclzip-') . '.tmp';

        // ----- Open the temporary file in write mode
        if (($v_zip_temp_fd = @fopen($v_zip_temp_name, 'wb')) == 0) {
            $this->privCloseFd();
            $p_archive_to_add->privCloseFd();

            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open temporary file \'' . $v_zip_temp_name . '\' in binary write mode');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Copy the files from the archive to the temporary file
        // TBC : Here I should better append the file and go back to erase the central dir
        $v_size = $v_central_dir['offset'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = fread($this->zip_fd, $v_read_size);
            @fwrite($v_zip_temp_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Copy the files from the archive_to_add into the temporary file
        $v_size = $v_central_dir_to_add['offset'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = fread($p_archive_to_add->zip_fd, $v_read_size);
            @fwrite($v_zip_temp_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Store the offset of the central dir
        $v_offset = @ftell($v_zip_temp_fd);

        // ----- Copy the block of file headers from the old archive
        $v_size = $v_central_dir['size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($this->zip_fd, $v_read_size);
            @fwrite($v_zip_temp_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Copy the block of file headers from the archive_to_add
        $v_size = $v_central_dir_to_add['size'];
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($p_archive_to_add->zip_fd, $v_read_size);
            @fwrite($v_zip_temp_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Merge the file comments
        $v_comment = $v_central_dir['comment'] . ' ' . $v_central_dir_to_add['comment'];

        // ----- Calculate the size of the (new) central header
        $v_size = @ftell($v_zip_temp_fd) - $v_offset;

        // ----- Swap the file descriptor
        // Here is a trick : I swap the temporary fd with the zip fd, in order to use
        // the following methods on the temporary fil and not the real archive fd
        $v_swap        = $this->zip_fd;
        $this->zip_fd  = $v_zip_temp_fd;
        $v_zip_temp_fd = $v_swap;

        // ----- Create the central dir footer
        if (($v_result = $this->privWriteCentralHeader($v_central_dir['entries'] + $v_central_dir_to_add['entries'], $v_size, $v_offset, $v_comment)) != 1) {
            $this->privCloseFd();
            $p_archive_to_add->privCloseFd();
            @fclose($v_zip_temp_fd);
            $this->zip_fd = null;

            // ----- Reset the file list
            unset($v_header_list);

            // ----- Return
            return $v_result;
        }

        // ----- Swap back the file descriptor
        $v_swap        = $this->zip_fd;
        $this->zip_fd  = $v_zip_temp_fd;
        $v_zip_temp_fd = $v_swap;

        // ----- Close
        $this->privCloseFd();
        $p_archive_to_add->privCloseFd();

        // ----- Close the temporary file
        @fclose($v_zip_temp_fd);

        // ----- Delete the zip file
        // TBC : I should test the result ...
        @unlink($this->zipname);

        // ----- Rename the temporary file
        // TBC : I should test the result ...
        //@rename($v_zip_temp_name, $this->zipname);
        PclZipUtilRename($v_zip_temp_name, $this->zipname);

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privDuplicate()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privDuplicate($p_archive_filename)
    {
        $v_result = 1;

        // ----- Look if the $p_archive_filename exists
        if (!is_file($p_archive_filename)) {

            // ----- Nothing to duplicate, so duplicate is a success.
            $v_result = 1;

            // ----- Return
            return $v_result;
        }

        // ----- Open the zip file
        if (($v_result = $this->privOpenFd('wb')) != 1) {
            // ----- Return
            return $v_result;
        }

        // ----- Open the temporary file in write mode
        if (($v_zip_temp_fd = @fopen($p_archive_filename, 'rb')) == 0) {
            $this->privCloseFd();

            PclZip::privErrorLog(PCLZIP_ERR_READ_OPEN_FAIL, 'Unable to open archive file \'' . $p_archive_filename . '\' in binary write mode');

            // ----- Return
            return PclZip::errorCode();
        }

        // ----- Copy the files from the archive to the temporary file
        // TBC : Here I should better append the file and go back to erase the central dir
        $v_size = filesize($p_archive_filename);
        while ($v_size != 0) {
            $v_read_size = ($v_size < PCLZIP_READ_BLOCK_SIZE ? $v_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = fread($v_zip_temp_fd, $v_read_size);
            @fwrite($this->zip_fd, $v_buffer, $v_read_size);
            $v_size -= $v_read_size;
        }

        // ----- Close
        $this->privCloseFd();

        // ----- Close the temporary file
        @fclose($v_zip_temp_fd);

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privErrorLog()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function privErrorLog($p_error_code = 0, $p_error_string = '')
    {
        if (PCLZIP_ERROR_EXTERNAL == 1) {
            PclError($p_error_code, $p_error_string);
        } else {
            $this->error_code   = $p_error_code;
            $this->error_string = $p_error_string;
        }
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privErrorReset()
    // Description :
    // Parameters :
    // --------------------------------------------------------------------------------
    public function privErrorReset()
    {
        if (PCLZIP_ERROR_EXTERNAL == 1) {
            PclErrorReset();
        } else {
            $this->error_code   = 0;
            $this->error_string = '';
        }
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privDisableMagicQuotes()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privDisableMagicQuotes()
    {
        $v_result = 1;

        // ----- Look if function exists
        if ((!function_exists("get_magic_quotes_runtime")) || (!function_exists("set_magic_quotes_runtime"))) {
            return $v_result;
        }

        // ----- Look if already done
        if ($this->magic_quotes_status != -1) {
            return $v_result;
        }

        // ----- Get and memorize the magic_quote value
        $this->magic_quotes_status = @get_magic_quotes_runtime();

        // ----- Disable magic_quotes
        if ($this->magic_quotes_status == 1) {
            @set_magic_quotes_runtime(0);
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------------
    // Function : privSwapBackMagicQuotes()
    // Description :
    // Parameters :
    // Return Values :
    // --------------------------------------------------------------------------------
    public function privSwapBackMagicQuotes()
    {
        $v_result = 1;

        // ----- Look if function exists
        if ((!function_exists("get_magic_quotes_runtime")) || (!function_exists("set_magic_quotes_runtime"))) {
            return $v_result;
        }

        // ----- Look if something to do
        if ($this->magic_quotes_status != -1) {
            return $v_result;
        }

        // ----- Swap back magic_quotes
        if ($this->magic_quotes_status == 1) {
            @set_magic_quotes_runtime($this->magic_quotes_status);
        }

        // ----- Return
        return $v_result;
    }
    // --------------------------------------------------------------------------------
}

// End of class
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilPathReduction()
// Description :
// Parameters :
// Return Values :
// --------------------------------------------------------------------------------
function PclZipUtilPathReduction($p_dir)
{
    $v_result = "";

    // ----- Look for not empty path
    if ($p_dir != "") {
        // ----- Explode path by directory names
        $v_list = explode("/", $p_dir);

        // ----- Study directories from last to first
        $v_skip = 0;
        for ($i = sizeof($v_list) - 1; $i >= 0; $i--) {
            // ----- Look for current path
            if ($v_list[$i] == ".") {
                // ----- Ignore this directory
                // Should be the first $i=0, but no check is done
            } elseif ($v_list[$i] == "..") {
                $v_skip++;
            } elseif ($v_list[$i] == "") {
                // ----- First '/' i.e. root slash
                if ($i == 0) {
                    $v_result = "/" . $v_result;
                    if ($v_skip > 0) {
                        // ----- It is an invalid path, so the path is not modified
                        // TBC
                        $v_result = $p_dir;
                        $v_skip   = 0;
                    }

                // ----- Last '/' i.e. indicates a directory
                } elseif ($i == (sizeof($v_list) - 1)) {
                    $v_result = $v_list[$i];

                // ----- Double '/' inside the path
                } else {
                    // ----- Ignore only the double '//' in path,
                    // but not the first and last '/'
                }
            } else {
                // ----- Look for item to skip
                if ($v_skip > 0) {
                    $v_skip--;
                } else {
                    $v_result = $v_list[$i] . ($i != (sizeof($v_list) - 1) ? "/" . $v_result : "");
                }
            }
        }

        // ----- Look for skip
        if ($v_skip > 0) {
            while ($v_skip > 0) {
                $v_result = '../' . $v_result;
                $v_skip--;
            }
        }
    }

    // ----- Return
    return $v_result;
}
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilPathInclusion()
// Description :
//   This function indicates if the path $p_path is under the $p_dir tree. Or,
//   said in an other way, if the file or sub-dir $p_path is inside the dir
//   $p_dir.
//   The function indicates also if the path is exactly the same as the dir.
//   This function supports path with duplicated '/' like '//', but does not
//   support '.' or '..' statements.
// Parameters :
// Return Values :
//   0 if $p_path is not inside directory $p_dir
//   1 if $p_path is inside directory $p_dir
//   2 if $p_path is exactly the same as $p_dir
// --------------------------------------------------------------------------------
function PclZipUtilPathInclusion($p_dir, $p_path)
{
    $v_result = 1;

    // ----- Look for path beginning by ./
    if (($p_dir == '.') || ((strlen($p_dir) >= 2) && (substr($p_dir, 0, 2) == './'))) {
        $p_dir = PclZipUtilTranslateWinPath(getcwd(), false) . '/' . substr($p_dir, 1);
    }
    if (($p_path == '.') || ((strlen($p_path) >= 2) && (substr($p_path, 0, 2) == './'))) {
        $p_path = PclZipUtilTranslateWinPath(getcwd(), false) . '/' . substr($p_path, 1);
    }

    // ----- Explode dir and path by directory separator
    $v_list_dir       = explode("/", $p_dir);
    $v_list_dir_size  = sizeof($v_list_dir);
    $v_list_path      = explode("/", $p_path);
    $v_list_path_size = sizeof($v_list_path);

    // ----- Study directories paths
    $i = 0;
    $j = 0;
    while (($i < $v_list_dir_size) && ($j < $v_list_path_size) && ($v_result)) {

        // ----- Look for empty dir (path reduction)
        if ($v_list_dir[$i] == '') {
            $i++;
            continue;
        }
        if ($v_list_path[$j] == '') {
            $j++;
            continue;
        }

        // ----- Compare the items
        if (($v_list_dir[$i] != $v_list_path[$j]) && ($v_list_dir[$i] != '') && ($v_list_path[$j] != '')) {
            $v_result = 0;
        }

        // ----- Next items
        $i++;
        $j++;
    }

    // ----- Look if everything seems to be the same
    if ($v_result) {
        // ----- Skip all the empty items
        while (($j < $v_list_path_size) && ($v_list_path[$j] == '')) {
            $j++;
        }
        while (($i < $v_list_dir_size) && ($v_list_dir[$i] == '')) {
            $i++;
        }

        if (($i >= $v_list_dir_size) && ($j >= $v_list_path_size)) {
            // ----- There are exactly the same
            $v_result = 2;
        } elseif ($i < $v_list_dir_size) {
            // ----- The path is shorter than the dir
            $v_result = 0;
        }
    }

    // ----- Return
    return $v_result;
}
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilCopyBlock()
// Description :
// Parameters :
//   $p_mode : read/write compression mode
//             0 : src & dest normal
//             1 : src gzip, dest normal
//             2 : src normal, dest gzip
//             3 : src & dest gzip
// Return Values :
// --------------------------------------------------------------------------------
function PclZipUtilCopyBlock($p_src, $p_dest, $p_size, $p_mode = 0)
{
    $v_result = 1;

    if ($p_mode == 0) {
        while ($p_size != 0) {
            $v_read_size = ($p_size < PCLZIP_READ_BLOCK_SIZE ? $p_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($p_src, $v_read_size);
            @fwrite($p_dest, $v_buffer, $v_read_size);
            $p_size -= $v_read_size;
        }
    } elseif ($p_mode == 1) {
        while ($p_size != 0) {
            $v_read_size = ($p_size < PCLZIP_READ_BLOCK_SIZE ? $p_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @gzread($p_src, $v_read_size);
            @fwrite($p_dest, $v_buffer, $v_read_size);
            $p_size -= $v_read_size;
        }
    } elseif ($p_mode == 2) {
        while ($p_size != 0) {
            $v_read_size = ($p_size < PCLZIP_READ_BLOCK_SIZE ? $p_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @fread($p_src, $v_read_size);
            @gzwrite($p_dest, $v_buffer, $v_read_size);
            $p_size -= $v_read_size;
        }
    } elseif ($p_mode == 3) {
        while ($p_size != 0) {
            $v_read_size = ($p_size < PCLZIP_READ_BLOCK_SIZE ? $p_size : PCLZIP_READ_BLOCK_SIZE);
            $v_buffer    = @gzread($p_src, $v_read_size);
            @gzwrite($p_dest, $v_buffer, $v_read_size);
            $p_size -= $v_read_size;
        }
    }

    // ----- Return
    return $v_result;
}
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilRename()
// Description :
//   This function tries to do a simple rename() function. If it fails, it
//   tries to copy the $p_src file in a new $p_dest file and then unlink the
//   first one.
// Parameters :
//   $p_src : Old filename
//   $p_dest : New filename
// Return Values :
//   1 on success, 0 on failure.
// --------------------------------------------------------------------------------
function PclZipUtilRename($p_src, $p_dest)
{
    $v_result = 1;

    // ----- Try to rename the files
    if (!@rename($p_src, $p_dest)) {

        // ----- Try to copy & unlink the src
        if (!@copy($p_src, $p_dest)) {
            $v_result = 0;
        } elseif (!@unlink($p_src)) {
            $v_result = 0;
        }
    }

    // ----- Return
    return $v_result;
}
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilOptionText()
// Description :
//   Translate option value in text. Mainly for debug purpose.
// Parameters :
//   $p_option : the option value.
// Return Values :
//   The option text value.
// --------------------------------------------------------------------------------
function PclZipUtilOptionText($p_option)
{

    $v_list = get_defined_constants();
    for (reset($v_list); $v_key = key($v_list); next($v_list)) {
        $v_prefix = substr($v_key, 0, 10);
        if ((($v_prefix == 'PCLZIP_OPT') || ($v_prefix == 'PCLZIP_CB_') || ($v_prefix == 'PCLZIP_ATT')) && ($v_list[$v_key] == $p_option)) {
            return $v_key;
        }
    }

    $v_result = 'Unknown';

    return $v_result;
}
// --------------------------------------------------------------------------------

// --------------------------------------------------------------------------------
// Function : PclZipUtilTranslateWinPath()
// Description :
//   Translate windows path by replacing '\' by '/' and optionally removing
//   drive letter.
// Parameters :
//   $p_path : path to translate.
//   $p_remove_disk_letter : true | false
// Return Values :
//   The path translated.
// --------------------------------------------------------------------------------
function PclZipUtilTranslateWinPath($p_path, $p_remove_disk_letter = true)
{
    if (stristr(php_uname(), 'windows')) {
        // ----- Look for potential disk letter
        if (($p_remove_disk_letter) && (($v_position = strpos($p_path, ':')) != false)) {
            $p_path = substr($p_path, $v_position + 1);
        }
        // ----- Change potential windows directory separator
        if ((strpos($p_path, '\\') > 0) || (substr($p_path, 0, 1) == '\\')) {
            $p_path = strtr($p_path, '\\', '/');
        }
    }

    return $p_path;
}
// --------------------------------------------------------------------------------

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

/**
 * Text
 */
class Text
{
    /**
     * Control characters array
     *
     * @var string[]
     */
    private static $controlCharacters = array();

    /**
     * Build control characters array
     */
    private static function buildControlCharacters()
    {
        for ($i = 0; $i <= 19; ++$i) {
            if ($i != 9 && $i != 10 && $i != 13) {
                $find = '_x' . sprintf('%04s', strtoupper(dechex($i))) . '_';
                $replace = chr($i);
                self::$controlCharacters[$find] = $replace;
            }
        }
    }

    /**
     * Convert from PHP control character to OpenXML escaped control character
     *
     * Excel 2007 team:
     * ----------------
     * That's correct, control characters are stored directly in the shared-strings table.
     * We do encode characters that cannot be represented in XML using the following escape sequence:
     * _xHHHH_ where H represents a hexadecimal character in the character's value...
     * So you could end up with something like _x0008_ in a string (either in a cell value (<v>)
     * element or in the shared string <t> element.
     *
     * @param  string $value Value to escape
     * @return string
     */
    public static function controlCharacterPHP2OOXML($value = '')
    {
        if (empty(self::$controlCharacters)) {
            self::buildControlCharacters();
        }

        return str_replace(array_values(self::$controlCharacters), array_keys(self::$controlCharacters), $value);
    }

    /**
     * Return a number formatted for being integrated in xml files
     * @param float $number
     * @param int $decimals
     * @return string
     */
    public static function numberFormat($number, $decimals)
    {
        return number_format($number, $decimals, '.', '');
    }

    /**
     * @param int $dec
     * @see http://stackoverflow.com/a/7153133/2235790
     * @author velcrow
     * @return string
     */
    public static function chr($dec)
    {
        if ($dec <= 0x7F) {
            return chr($dec);
        }
        if ($dec <= 0x7FF) {
            return chr(($dec >> 6) + 192) . chr(($dec & 63) + 128);
        }
        if ($dec <= 0xFFFF) {
            return chr(($dec >> 12) + 224) . chr((($dec >> 6) & 63) + 128) . chr(($dec & 63) + 128);
        }
        if ($dec <= 0x1FFFFF) {
            return chr(($dec >> 18) + 240) . chr((($dec >> 12) & 63) + 128) . chr((($dec >> 6) & 63) + 128) . chr(($dec & 63) + 128);
        }

        return '';
    }

    /**
     * Convert from OpenXML escaped control character to PHP control character
     *
     * @param string $value Value to unescape
     * @return string
     */
    public static function controlCharacterOOXML2PHP($value = '')
    {
        if (empty(self::$controlCharacters)) {
            self::buildControlCharacters();
        }

        return str_replace(array_keys(self::$controlCharacters), array_values(self::$controlCharacters), $value);
    }

    /**
     * Check if a string contains UTF-8 data
     *
     * @param string $value
     * @return bool
     */
    public static function isUTF8($value = '')
    {
        return is_string($value) && ($value === '' || preg_match('/^./su', $value) == 1);
    }

    /**
     * Return UTF8 encoded value
     *
     * @param string $value
     * @return string
     */
    public static function toUTF8($value = '')
    {
        if (!is_null($value) && !self::isUTF8($value)) {
            $value = utf8_encode($value);
        }

        return $value;
    }

    /**
     * Returns unicode from UTF8 text
     *
     * The function is splitted to reduce cyclomatic complexity
     *
     * @param string $text UTF8 text
     * @return string Unicode text
     * @since 0.11.0
     */
    public static function toUnicode($text)
    {
        return self::unicodeToEntities(self::utf8ToUnicode($text));
    }

    /**
     * Returns unicode array from UTF8 text
     *
     * @param string $text UTF8 text
     * @return array
     * @since 0.11.0
     * @see http://www.randomchaos.com/documents/?source=php_and_unicode
     */
    public static function utf8ToUnicode($text)
    {
        $unicode = array();
        $values = array();
        $lookingFor = 1;

        // Gets unicode for each character
        for ($i = 0; $i < strlen($text); $i++) {
            $thisValue = ord($text[$i]);
            if ($thisValue < 128) {
                $unicode[] = $thisValue;
            } else {
                if (count($values) == 0) {
                    $lookingFor = $thisValue < 224 ? 2 : 3;
                }
                $values[] = $thisValue;
                if (count($values) == $lookingFor) {
                    if ($lookingFor == 3) {
                        $number = (($values[0] % 16) * 4096) + (($values[1] % 64) * 64) + ($values[2] % 64);
                    } else {
                        $number = (($values[0] % 32) * 64) + ($values[1] % 64);
                    }
                    $unicode[] = $number;
                    $values = array();
                    $lookingFor = 1;
                }
            }
        }

        return $unicode;
    }

    /**
     * Returns entites from unicode array
     *
     * @param array $unicode
     * @return string
     * @since 0.11.0
     * @see http://www.randomchaos.com/documents/?source=php_and_unicode
     */
    private static function unicodeToEntities($unicode)
    {
        $entities = '';

        foreach ($unicode as $value) {
            if ($value != 65279) {
                $entities .= $value > 127 ? '\uc0{\u' . $value . '}' : chr($value);
            }
        }

        return $entities;
    }

    /**
     * Return name without underscore for < 0.10.0 variable name compatibility
     *
     * @param string $value
     * @return string
     */
    public static function removeUnderscorePrefix($value)
    {
        if (!is_null($value)) {
            if (substr($value, 0, 1) == '_') {
                $value = substr($value, 1);
            }
        }

        return $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/XMLReader.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

/**
 * XML Reader wrapper
 *
 * @since   0.2.1
 */
class XMLReader
{
    /**
     * DOMDocument object
     *
     * @var \DOMDocument
     */
    private $dom = null;

    /**
     * DOMXpath object
     *
     * @var \DOMXpath
     */
    private $xpath = null;

    /**
     * Get DOMDocument from ZipArchive
     *
     * @param string $zipFile
     * @param string $xmlFile
     * @throws \Exception
     * @return \DOMDocument|false
     */
    public function getDomFromZip($zipFile, $xmlFile)
    {
        if (file_exists($zipFile) === false) {
            throw new \Exception('Cannot find archive file.');
        }

        $zip = new \ZipArchive();
        $zip->open($zipFile);
        $content = $zip->getFromName($xmlFile);
        $zip->close();

        if ($content === false) {
            return false;
        }

        return $this->getDomFromString($content);
    }

    /**
     * Get DOMDocument from content string
     *
     * @param string $content
     * @return \DOMDocument
     */
    public function getDomFromString($content)
    {
        if (\PHP_VERSION_ID < 80000) {
            $originalLibXMLEntityValue = libxml_disable_entity_loader(true);
        }
        $this->dom = new \DOMDocument();
        $this->dom->loadXML($content);
        if (\PHP_VERSION_ID < 80000) {
            libxml_disable_entity_loader($originalLibXMLEntityValue);
        }

        return $this->dom;
    }

    /**
     * Get elements
     *
     * @param string $path
     * @param \DOMElement $contextNode
     * @return \DOMNodeList
     */
    public function getElements($path, \DOMElement $contextNode = null)
    {
        if ($this->dom === null) {
            return array();
        }
        if ($this->xpath === null) {
            $this->xpath = new \DOMXpath($this->dom);
        }

        if (is_null($contextNode)) {
            return $this->xpath->query($path);
        }

        return $this->xpath->query($path, $contextNode);
    }

    /**
     * Registers the namespace with the DOMXPath object
     *
     * @param string $prefix The prefix
     * @param string $namespaceURI The URI of the namespace
     * @throws \InvalidArgumentException If called before having loaded the DOM document
     * @return bool true on success or false on failure
     */
    public function registerNamespace($prefix, $namespaceURI)
    {
        if ($this->dom === null) {
            throw new \InvalidArgumentException('Dom needs to be loaded before registering a namespace');
        }
        if ($this->xpath === null) {
            $this->xpath = new \DOMXpath($this->dom);
        }

        return $this->xpath->registerNamespace($prefix, $namespaceURI);
    }

    /**
     * Get element
     *
     * @param string $path
     * @param \DOMElement $contextNode
     * @return \DOMElement|null
     */
    public function getElement($path, \DOMElement $contextNode = null)
    {
        $elements = $this->getElements($path, $contextNode);
        if ($elements->length > 0) {
            return $elements->item(0);
        }

        return null;
    }

    /**
     * Get element attribute
     *
     * @param string $attribute
     * @param \DOMElement $contextNode
     * @param string $path
     * @return string|null
     */
    public function getAttribute($attribute, \DOMElement $contextNode = null, $path = null)
    {
        $return = null;
        if ($path !== null) {
            $elements = $this->getElements($path, $contextNode);
            if ($elements->length > 0) {
                /** @var \DOMElement $node Type hint */
                $node = $elements->item(0);
                $return = $node->getAttribute($attribute);
            }
        } else {
            if ($contextNode !== null) {
                $return = $contextNode->getAttribute($attribute);
            }
        }

        return ($return == '') ? null : $return;
    }

    /**
     * Get element value
     *
     * @param string $path
     * @param \DOMElement $contextNode
     * @return string|null
     */
    public function getValue($path, \DOMElement $contextNode = null)
    {
        $elements = $this->getElements($path, $contextNode);
        if ($elements->length > 0) {
            return $elements->item(0)->nodeValue;
        }

        return null;
    }

    /**
     * Count elements
     *
     * @param string $path
     * @param \DOMElement $contextNode
     * @return int
     */
    public function countElements($path, \DOMElement $contextNode = null)
    {
        $elements = $this->getElements($path, $contextNode);

        return $elements->length;
    }

    /**
     * Element exists
     *
     * @param string $path
     * @param \DOMElement $contextNode
     * @return bool
     */
    public function elementExists($path, \DOMElement $contextNode = null)
    {
        return $this->getElements($path, $contextNode)->length > 0;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/XMLWriter.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

/**
 * XMLWriter
 *
 * @method bool endElement()
 * @method mixed flush(bool $empty = null)
 * @method bool openMemory()
 * @method string outputMemory(bool $flush = null)
 * @method bool setIndent(bool $indent)
 * @method bool startDocument(string $version = 1.0, string $encoding = null, string $standalone = null)
 * @method bool startElement(string $name)
 * @method bool text(string $content)
 * @method bool writeCData(string $content)
 * @method bool writeComment(string $content)
 * @method bool writeElement(string $name, string $content = null)
 * @method bool writeRaw(string $content)
 */
class XMLWriter extends \XMLWriter
{
    /** Temporary storage method */
    const STORAGE_MEMORY = 1;
    const STORAGE_DISK = 2;

    /**
     * Temporary filename
     *
     * @var string
     */
    private $tempFileName = '';

    /**
     * Create a new \PhpOffice\PhpWord\Shared\XMLWriter instance
     *
     * @param int $pTemporaryStorage Temporary storage location
     * @param string $pTemporaryStorageDir Temporary storage folder
     * @param bool $compatibility
     */
    public function __construct($pTemporaryStorage = self::STORAGE_MEMORY, $pTemporaryStorageDir = null, $compatibility = false)
    {
        // Open temporary storage
        if ($pTemporaryStorage == self::STORAGE_MEMORY) {
            $this->openMemory();
        } else {
            if (!is_dir($pTemporaryStorageDir)) {
                $pTemporaryStorageDir = sys_get_temp_dir();
            }
            // Create temporary filename
            $this->tempFileName = @tempnam($pTemporaryStorageDir, 'xml');

            // Open storage
            $this->openUri($this->tempFileName);
        }

        if ($compatibility) {
            $this->setIndent(false);
            $this->setIndentString('');
        } else {
            $this->setIndent(true);
            $this->setIndentString('  ');
        }
    }

    /**
     * Destructor
     */
    public function __destruct()
    {
        // Unlink temporary files
        if (empty($this->tempFileName)) {
            return;
        }
        if (PHP_OS != 'WINNT' && @unlink($this->tempFileName) === false) {
            throw new \Exception('The file ' . $this->tempFileName . ' could not be deleted.');
        }
    }

    /**
     * Get written data
     *
     * @return string
     */
    public function getData()
    {
        if ($this->tempFileName == '') {
            return $this->outputMemory(true);
        }

        $this->flush();

        return file_get_contents($this->tempFileName);
    }

    /**
     * Write simple element and attribute(s) block
     *
     * There are two options:
     * 1. If the `$attributes` is an array, then it's an associative array of attributes
     * 2. If not, then it's a simple attribute-value pair
     *
     * @param string $element
     * @param string|array $attributes
     * @param string $value
     */
    public function writeElementBlock($element, $attributes, $value = null)
    {
        $this->startElement($element);
        if (!is_array($attributes)) {
            $attributes = array($attributes => $value);
        }
        foreach ($attributes as $attribute => $value) {
            $this->writeAttribute($attribute, $value);
        }
        $this->endElement();
    }

    /**
     * Write element if ...
     *
     * @param bool $condition
     * @param string $element
     * @param string $attribute
     * @param mixed $value
     */
    public function writeElementIf($condition, $element, $attribute = null, $value = null)
    {
        if ($condition == true) {
            if (is_null($attribute)) {
                $this->writeElement($element, $value);
            } else {
                $this->startElement($element);
                $this->writeAttribute($attribute, $value);
                $this->endElement();
            }
        }
    }

    /**
     * Write attribute if ...
     *
     * @param bool $condition
     * @param string $attribute
     * @param mixed $value
     */
    public function writeAttributeIf($condition, $attribute, $value)
    {
        if ($condition == true) {
            $this->writeAttribute($attribute, $value);
        }
    }

    /**
     * @param string $name
     * @param mixed $value
     * @return bool
     */
    public function writeAttribute($name, $value)
    {
        if (is_float($value)) {
            $value = json_encode($value);
        }

        return parent::writeAttribute($name, $value);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Shared/ZipArchive.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Shared;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Settings;

/**
 * ZipArchive wrapper
 *
 * Wraps zip archive functionality of PHP ZipArchive and PCLZip. PHP ZipArchive
 * properties and methods are bypassed and used as the model for the PCLZip
 * emulation. Only needed PHP ZipArchive features are implemented.
 *
 * @method  bool addFile(string $filename, string $localname = null)
 * @method  bool addFromString(string $localname, string $contents)
 * @method  string getNameIndex(int $index)
 * @method  int locateName(string $name)
 *
 * @since   0.10.0
 */
class ZipArchive
{
    /** @const int Flags for open method */
    const CREATE = 1; // Emulate \ZipArchive::CREATE
    const OVERWRITE = 8; // Emulate \ZipArchive::OVERWRITE

    /**
     * Number of files (emulate ZipArchive::$numFiles)
     *
     * @var int
     */
    public $numFiles = 0;

    /**
     * Archive filename (emulate ZipArchive::$filename)
     *
     * @var string
     */
    public $filename;

    /**
     * Temporary storage directory
     *
     * @var string
     */
    private $tempDir;

    /**
     * Internal zip archive object
     *
     * @var \ZipArchive|\PclZip
     */
    private $zip;

    /**
     * Use PCLZip (default behaviour)
     *
     * @var bool
     */
    private $usePclzip = true;

    /**
     * Create new instance
     */
    public function __construct()
    {
        $this->usePclzip = (Settings::getZipClass() != 'ZipArchive');
        if ($this->usePclzip) {
            if (!defined('PCLZIP_TEMPORARY_DIR')) {
                define('PCLZIP_TEMPORARY_DIR', Settings::getTempDir() . '/');
            }
            require_once 'PCLZip/pclzip.lib.php';
        }
    }

    /**
     * Catch function calls: pass to ZipArchive or PCLZip
     *
     * `call_user_func_array` can only used for public function, hence the `public` in all `pcl...` methods
     *
     * @param mixed $function
     * @param mixed $args
     * @return mixed
     */
    public function __call($function, $args)
    {
        // Set object and function
        $zipFunction = $function;
        if (!$this->usePclzip) {
            $zipObject = $this->zip;
        } else {
            $zipObject = $this;
            $zipFunction = "pclzip{$zipFunction}";
        }

        // Run function
        $result = false;
        if (method_exists($zipObject, $zipFunction)) {
            $result = @call_user_func_array(array($zipObject, $zipFunction), $args);
        }

        return $result;
    }

    /**
     * Open a new zip archive
     *
     * @param string $filename The file name of the ZIP archive to open
     * @param int $flags The mode to use to open the archive
     * @return bool
     */
    public function open($filename, $flags = null)
    {
        $result = true;
        $this->filename = $filename;
        $this->tempDir = Settings::getTempDir();

        if (!$this->usePclzip) {
            $zip = new \ZipArchive();
            $result = $zip->open($this->filename, $flags);

            // Scrutizer will report the property numFiles does not exist
            // See https://github.com/scrutinizer-ci/php-analyzer/issues/190
            $this->numFiles = $zip->numFiles;
        } else {
            $zip = new \PclZip($this->filename);
            $zipContent = $zip->listContent();
            $this->numFiles = is_array($zipContent) ? count($zipContent) : 0;
        }
        $this->zip = $zip;

        return $result;
    }

    /**
     * Close the active archive
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return bool
     *
     * @codeCoverageIgnore Can't find any test case. Uncomment when found.
     */
    public function close()
    {
        if (!$this->usePclzip) {
            if ($this->zip->close() === false) {
                throw new Exception("Could not close zip file {$this->filename}: ");
            }
        }

        return true;
    }

    /**
     * Extract the archive contents (emulate \ZipArchive)
     *
     * @param string $destination
     * @param string|array $entries
     * @return bool
     * @since 0.10.0
     */
    public function extractTo($destination, $entries = null)
    {
        if (!is_dir($destination)) {
            return false;
        }

        if (!$this->usePclzip) {
            return $this->zip->extractTo($destination, $entries);
        }

        return $this->pclzipExtractTo($destination, $entries);
    }

    /**
     * Extract file from archive by given file name (emulate \ZipArchive)
     *
     * @param  string $filename Filename for the file in zip archive
     * @return string $contents File string contents
     */
    public function getFromName($filename)
    {
        if (!$this->usePclzip) {
            $contents = $this->zip->getFromName($filename);
            if ($contents === false) {
                $filename = substr($filename, 1);
                $contents = $this->zip->getFromName($filename);
            }
        } else {
            $contents = $this->pclzipGetFromName($filename);
        }

        return $contents;
    }

    /**
     * Add a new file to the zip archive (emulate \ZipArchive)
     *
     * @param string $filename Directory/Name of the file to add to the zip archive
     * @param string $localname Directory/Name of the file added to the zip
     * @return bool
     */
    public function pclzipAddFile($filename, $localname = null)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;

        // Bugfix GH-261 https://github.com/PHPOffice/PHPWord/pull/261
        $realpathFilename = realpath($filename);
        if ($realpathFilename !== false) {
            $filename = $realpathFilename;
        }

        $filenameParts = pathinfo($filename);
        $localnameParts = pathinfo($localname);

        // To Rename the file while adding it to the zip we
        //   need to create a temp file with the correct name
        $tempFile = false;
        if ($filenameParts['basename'] != $localnameParts['basename']) {
            $tempFile = true; // temp file created
            $temppath = $this->tempDir . DIRECTORY_SEPARATOR . $localnameParts['basename'];
            copy($filename, $temppath);
            $filename = $temppath;
            $filenameParts = pathinfo($temppath);
        }

        $pathRemoved = $filenameParts['dirname'];
        $pathAdded = $localnameParts['dirname'];

        if (!$this->usePclzip) {
            $pathAdded = $pathAdded . '/' . ltrim(str_replace('\\', '/', substr($filename, strlen($pathRemoved))), '/');
            //$res = $zip->addFile($filename, $pathAdded);
            $res = $zip->addFromString($pathAdded, file_get_contents($filename));       // addFile can't use subfolders in some cases
        } else {
            $res = $zip->add($filename, PCLZIP_OPT_REMOVE_PATH, $pathRemoved, PCLZIP_OPT_ADD_PATH, $pathAdded);
        }

        if ($tempFile) {
            // Remove temp file, if created
            unlink($this->tempDir . DIRECTORY_SEPARATOR . $localnameParts['basename']);
        }

        return $res != 0;
    }

    /**
     * Add a new file to the zip archive from a string of raw data (emulate \ZipArchive)
     *
     * @param string $localname Directory/Name of the file to add to the zip archive
     * @param string $contents String of data to add to the zip archive
     * @return bool
     */
    public function pclzipAddFromString($localname, $contents)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;
        $filenameParts = pathinfo($localname);

        // Write $contents to a temp file
        $handle = fopen($this->tempDir . DIRECTORY_SEPARATOR . $filenameParts['basename'], 'wb');
        fwrite($handle, $contents);
        fclose($handle);

        // Add temp file to zip
        $filename = $this->tempDir . DIRECTORY_SEPARATOR . $filenameParts['basename'];
        $pathRemoved = $this->tempDir;
        $pathAdded = $filenameParts['dirname'];

        $res = $zip->add($filename, PCLZIP_OPT_REMOVE_PATH, $pathRemoved, PCLZIP_OPT_ADD_PATH, $pathAdded);

        // Remove temp file
        @unlink($this->tempDir . DIRECTORY_SEPARATOR . $filenameParts['basename']);

        return $res != 0;
    }

    /**
     * Extract the archive contents (emulate \ZipArchive)
     *
     * @param string $destination
     * @param string|array $entries
     * @return bool
     * @since 0.10.0
     */
    public function pclzipExtractTo($destination, $entries = null)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;

        // Extract all files
        if (is_null($entries)) {
            $result = $zip->extract(PCLZIP_OPT_PATH, $destination);

            return $result > 0;
        }

        // Extract by entries
        if (!is_array($entries)) {
            $entries = array($entries);
        }
        foreach ($entries as $entry) {
            $entryIndex = $this->locateName($entry);
            $result = $zip->extractByIndex($entryIndex, PCLZIP_OPT_PATH, $destination);
            if ($result <= 0) {
                return false;
            }
        }

        return true;
    }

    /**
     * Extract file from archive by given file name (emulate \ZipArchive)
     *
     * @param  string $filename Filename for the file in zip archive
     * @return string $contents File string contents
     */
    public function pclzipGetFromName($filename)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;
        $listIndex = $this->pclzipLocateName($filename);
        $contents = false;

        if ($listIndex !== false) {
            $extracted = $zip->extractByIndex($listIndex, PCLZIP_OPT_EXTRACT_AS_STRING);
        } else {
            $filename = substr($filename, 1);
            $listIndex = $this->pclzipLocateName($filename);
            $extracted = $zip->extractByIndex($listIndex, PCLZIP_OPT_EXTRACT_AS_STRING);
        }
        if ((is_array($extracted)) && ($extracted != 0)) {
            $contents = $extracted[0]['content'];
        }

        return $contents;
    }

    /**
     * Returns the name of an entry using its index (emulate \ZipArchive)
     *
     * @param int $index
     * @return string|bool
     * @since 0.10.0
     */
    public function pclzipGetNameIndex($index)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;
        $list = $zip->listContent();
        if (isset($list[$index])) {
            return $list[$index]['filename'];
        }

        return false;
    }

    /**
     * Returns the index of the entry in the archive (emulate \ZipArchive)
     *
     * @param string $filename Filename for the file in zip archive
     * @return int
     */
    public function pclzipLocateName($filename)
    {
        /** @var \PclZip $zip Type hint */
        $zip = $this->zip;
        $list = $zip->listContent();
        $listCount = count($list);
        $listIndex = -1;
        for ($i = 0; $i < $listCount; ++$i) {
            if (strtolower($list[$i]['filename']) == strtolower($filename) ||
                strtolower($list[$i]['stored_filename']) == strtolower($filename)) {
                $listIndex = $i;
                break;
            }
        }

        return ($listIndex > -1) ? $listIndex : false;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/Border.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Border Styles.
 *
 * @since 0.18.0
 *
 * @see  http://www.datypic.com/sc/ooxml/t-w_ST_Border.html
 */
final class Border extends AbstractEnum
{
    const SINGLE = 'single'; //A single line
    const DASH_DOT_STROKED = 'dashDotStroked'; //A line with a series of alternating thin and thick strokes
    const DASHED = 'dashed'; //A dashed line
    const DASH_SMALL_GAP = 'dashSmallGap'; //A dashed line with small gaps
    const DOT_DASH = 'dotDash'; //A line with alternating dots and dashes
    const DOT_DOT_DASH = 'dotDotDash'; //A line with a repeating dot - dot - dash sequence
    const DOTTED = 'dotted'; //A dotted line
    const DOUBLE = 'double'; //A double line
    const DOUBLE_WAVE = 'doubleWave'; //A double wavy line
    const INSET = 'inset'; //An inset set of lines
    const NIL = 'nil'; //No border
    const NONE = 'none'; //No border
    const OUTSET = 'outset'; //An outset set of lines
    const THICK = 'thick'; //A single line
    const THICK_THIN_LARGE_GAP = 'thickThinLargeGap'; //A thick line contained within a thin line with a large-sized intermediate gap
    const THICK_THIN_MEDIUM_GAP = 'thickThinMediumGap'; //A thick line contained within a thin line with a medium-sized intermediate gap
    const THICK_THIN_SMALL_GAP = 'thickThinSmallGap'; //A thick line contained within a thin line with a small intermediate gap
    const THIN_THICK_LARGE_GAP = 'thinThickLargeGap'; //A thin line contained within a thick line with a large-sized intermediate gap
    const THIN_THICK_MEDIUM_GAP = 'thinThickMediumGap'; //A thick line contained within a thin line with a medium-sized intermediate gap
    const THIN_THICK_SMALL_GAP = 'thinThickSmallGap'; //A thick line contained within a thin line with a small intermediate gap
    const THIN_THICK_THINLARGE_GAP = 'thinThickThinLargeGap'; //A thin-thick-thin line with a large gap
    const THIN_THICK_THIN_MEDIUM_GAP = 'thinThickThinMediumGap'; //A thin-thick-thin line with a medium gap
    const THIN_THICK_THIN_SMALL_GAP = 'thinThickThinSmallGap'; //A thin-thick-thin line with a small gap
    const THREE_D_EMBOSS = 'threeDEmboss'; //A three-staged gradient line, getting darker towards the paragraph
    const THREE_D_ENGRAVE = 'threeDEngrave'; //A three-staged gradient like, getting darker away from the paragraph
    const TRIPLE = 'triple'; //A triple line
    const WAVE = 'wave'; //A wavy line
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/DocProtect.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Document Protection Types
 *
 * @since 0.14.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_DocProtect.html
 */
final class DocProtect extends AbstractEnum
{
    /**
     * No Editing Restrictions
     */
    const NONE = 'none';

    /**
     * Allow No Editing
     */
    const READ_ONLY = 'readOnly';

    /**
     * Allow Editing of Comments
     */
    const COMMENTS = 'comments';

    /**
     * Allow Editing With Revision Tracking
     */
    const TRACKED_CHANGES = 'trackedChanges';

    /**
     * Allow Editing of Form Fields
     */
    const FORMS = 'forms';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/Jc.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Horizontal Alignment Type.
 *
 * Introduced in 1st Edition of ECMA-376. Initially it was intended to align paragraphs and tables.
 * Since ISO/IEC-29500:2008 the type must not be used for table alignment.
 *
 * @since 0.13.0
 *
 * @see \PhpOffice\PhpWord\SimpleType\JcTable For table alignment modes available since ISO/IEC-29500:2008.
 * @see  http://www.datypic.com/sc/ooxml/t-w_ST_Jc.html
 */
final class Jc extends AbstractEnum
{
    const START = 'start';
    const CENTER = 'center';
    const END = 'end';
    const BOTH = 'both';
    const MEDIUM_KASHIDA = 'mediumKashida';
    const DISTRIBUTE = 'distribute';
    const NUM_TAB = 'numTab';
    const HIGH_KASHIDA = 'highKashida';
    const LOW_KASHIDA = 'lowKashida';
    const THAI_DISTRIBUTE = 'thaiDistribute';

    /**
     * Kept for compatibility with 1st edition of ECMA-376 standard.
     * Microsoft Word 2007 and WPS Writer 2016 still rely on it.
     *
     * @deprecated 0.13.0 For documents based on ISO/IEC 29500:2008 and later use `START` instead.
     */
    const LEFT = 'left';
    /**
     * Kept for compatibility with 1st edition of ECMA-376 standard.
     * Microsoft Word 2007 and WPS Writer 2016 still rely on it.
     *
     * @deprecated 0.13.0 For documents based on ISO/IEC 29500:2008 and later use `END` instead.
     */
    const RIGHT = 'right';
    /**
     * Kept for compatibility with 1st edition of ECMA-376 standard.
     * Microsoft Word 2007 and WPS Writer 2016 still rely on it.
     *
     * @deprecated 0.13.0 For documents based on ISO/IEC 29500:2008 and later use `BOTH` instead.
     */
    const JUSTIFY = 'justify';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/JcTable.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Table Alignment Type.
 *
 * Introduced in ISO/IEC-29500:2008.
 *
 * @since 0.13.0
 */
final class JcTable extends AbstractEnum
{
    const START = 'start';
    const CENTER = 'center';
    const END = 'end';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/LineSpacingRule.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Line Spacing Rule
 *
 * @since 0.14.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_LineSpacingRule.html
 */
final class LineSpacingRule extends AbstractEnum
{
    /**
     * Automatically Determined Line Height
     */
    const AUTO = 'auto';

    /**
     * Exact Line Height
     */
    const EXACT = 'exact';

    /**
     * Minimum Line Height
     */
    const AT_LEAST = 'atLeast';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/NumberFormat.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Numbering Format.
 *
 * @since 0.14.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_NumberFormat.html.
 */
final class NumberFormat extends AbstractEnum
{
    //Decimal Numbers
    const DECIMAL = 'decimal';
    //Uppercase Roman Numerals
    const UPPER_ROMAN = 'upperRoman';
    //Lowercase Roman Numerals
    const LOWER_ROMAN = 'lowerRoman';
    //Uppercase Latin Alphabet
    const UPPER_LETTER = 'upperLetter';
    //Lowercase Latin Alphabet
    const LOWER_LETTER = 'lowerLetter';
    //Ordinal
    const ORDINAL = 'ordinal';
    //Cardinal Text
    const CARDINAL_TEXT = 'cardinalText';
    //Ordinal Text
    const ORDINAL_TEXT = 'ordinalText';
    //Hexadecimal Numbering
    const HEX = 'hex';
    //Chicago Manual of Style
    const CHICAGO = 'chicago';
    //Ideographs
    const IDEOGRAPH_DIGITAL = 'ideographDigital';
    //Japanese Counting System
    const JAPANESE_COUNTING = 'japaneseCounting';
    //AIUEO Order Hiragana
    const AIUEO = 'aiueo';
    //Iroha Ordered Katakana
    const IROHA = 'iroha';
    //Double Byte Arabic Numerals
    const DECIMAL_FULL_WIDTH = 'decimalFullWidth';
    //Single Byte Arabic Numerals
    const DECIMAL_HALF_WIDTH = 'decimalHalfWidth';
    //Japanese Legal Numbering
    const JAPANESE_LEGAL = 'japaneseLegal';
    //Japanese Digital Ten Thousand Counting System
    const JAPANESE_DIGITAL_TEN_THOUSAND = 'japaneseDigitalTenThousand';
    //Decimal Numbers Enclosed in a Circle
    const DECIMAL_ENCLOSED_CIRCLE = 'decimalEnclosedCircle';
    //Double Byte Arabic Numerals Alternate
    const DECIMAL_FULL_WIDTH2 = 'decimalFullWidth2';
    //Full-Width AIUEO Order Hiragana
    const AIUEO_FULL_WIDTH = 'aiueoFullWidth';
    //Full-Width Iroha Ordered Katakana
    const IROHA_FULL_WIDTH = 'irohaFullWidth';
    //Initial Zero Arabic Numerals
    const DECIMAL_ZERO = 'decimalZero';
    //Bullet
    const BULLET = 'bullet';
    //Korean Ganada Numbering
    const GANADA = 'ganada';
    //Korean Chosung Numbering
    const CHOSUNG = 'chosung';
    //Decimal Numbers Followed by a Period
    const DECIMAL_ENCLOSED_FULL_STOP = 'decimalEnclosedFullstop';
    //Decimal Numbers Enclosed in Parenthesis
    const DECIMAL_ENCLOSED_PAREN = 'decimalEnclosedParen';
    //Decimal Numbers Enclosed in a Circle
    const DECIMAL_ENCLOSED_CIRCLE_CHINESE = 'decimalEnclosedCircleChinese';
    //Ideographs Enclosed in a Circle
    const IDEOGRAPHENCLOSEDCIRCLE = 'ideographEnclosedCircle';
    //Traditional Ideograph Format
    const IDEOGRAPH_TRADITIONAL = 'ideographTraditional';
    //Zodiac Ideograph Format
    const IDEOGRAPH_ZODIAC = 'ideographZodiac';
    //Traditional Zodiac Ideograph Format
    const IDEOGRAPH_ZODIAC_TRADITIONAL = 'ideographZodiacTraditional';
    //Taiwanese Counting System
    const TAIWANESE_COUNTING = 'taiwaneseCounting';
    //Traditional Legal Ideograph Format
    const IDEOGRAPH_LEGAL_TRADITIONAL = 'ideographLegalTraditional';
    //Taiwanese Counting Thousand System
    const TAIWANESE_COUNTING_THOUSAND = 'taiwaneseCountingThousand';
    //Taiwanese Digital Counting System
    const TAIWANESE_DIGITAL = 'taiwaneseDigital';
    //Chinese Counting System
    const CHINESE_COUNTING = 'chineseCounting';
    //Chinese Legal Simplified Format
    const CHINESE_LEGAL_SIMPLIFIED = 'chineseLegalSimplified';
    //Chinese Counting Thousand System
    const CHINESE_COUNTING_THOUSAND = 'chineseCountingThousand';
    //Korean Digital Counting System
    const KOREAN_DIGITAL = 'koreanDigital';
    //Korean Counting System
    const KOREAN_COUNTING = 'koreanCounting';
    //Korean Legal Numbering
    const KOREAN_LEGAL = 'koreanLegal';
    //Korean Digital Counting System Alternate
    const KOREAN_DIGITAL2 = 'koreanDigital2';
    //Vietnamese Numerals
    const VIETNAMESE_COUNTING = 'vietnameseCounting';
    //Lowercase Russian Alphabet
    const RUSSIAN_LOWER = 'russianLower';
    //Uppercase Russian Alphabet
    const RUSSIAN_UPPER = 'russianUpper';
    //No Numbering
    const NONE = 'none';
    //Number With Dashes
    const NUMBER_IN_DASH = 'numberInDash';
    //Hebrew Numerals
    const HEBREW1 = 'hebrew1';
    //Hebrew Alphabet
    const HEBREW2 = 'hebrew2';
    //Arabic Alphabet
    const ARABIC_ALPHA = 'arabicAlpha';
    //Arabic Abjad Numerals
    const ARABIC_ABJAD = 'arabicAbjad';
    //Hindi Vowels
    const HINDI_VOWELS = 'hindiVowels';
    //Hindi Consonants
    const HINDI_CONSONANTS = 'hindiConsonants';
    //Hindi Numbers
    const HINDI_NUMBERS = 'hindiNumbers';
    //Hindi Counting System
    const HINDI_COUNTING = 'hindiCounting';
    //Thai Letters
    const THAI_LETTERS = 'thaiLetters';
    //Thai Numerals
    const THAI_NUMBERS = 'thaiNumbers';
    //Thai Counting System
    const THAI_COUNTING = 'thaiCounting';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/TblWidth.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Table Width Units
 *
 * @since 0.15.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_TblWidth.html
 */
final class TblWidth extends AbstractEnum
{
    //No Width
    const NIL = 'nil';

    //Automatically Determined Width
    const AUTO = 'auto';

    //Width in Fiftieths of a Percent
    const PERCENT = 'pct';

    //Width in Twentieths of a Point
    const TWIP = 'dxa';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/TextAlignment.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Magnification Preset Values
 *
 * @since 0.14.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_TextAlignment.html
 */
final class TextAlignment extends AbstractEnum
{
    //Align Text at Top
    const TOP = 'top';

    //Align Text at Center
    const CENTER = 'center';

    //Align Text at Baseline
    const BASELINE = 'baseline';

    //Align Text at Bottom
    const BOTTOM = 'bottom';

    //Automatically Determine Alignment
    const AUTO = 'auto';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/VerticalJc.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Vertical Alignment Type.
 *
 * Introduced in ISO/IEC-29500:2008.
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_VerticalJc.html
 * @since 0.17.0
 */
final class VerticalJc extends AbstractEnum
{
    const TOP = 'top';
    const CENTER = 'center';
    const BOTH = 'both';
    const BOTTOM = 'bottom';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/SimpleType/Zoom.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\SimpleType;

use PhpOffice\PhpWord\Shared\AbstractEnum;

/**
 * Magnification Preset Values
 *
 * @since 0.14.0
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_ST_Zoom.html
 */
final class Zoom extends AbstractEnum
{
    //No Preset Magnification
    const NONE = 'none';

    //Display One Full Page
    const FULL_PAGE = 'fullPage';

    //Display Page Width
    const BEST_FIT = 'bestFit';

    //Display Text Width
    const TEXT_FIT = 'textFit';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/AbstractStyle.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\Shared\Text;

/**
 * Abstract style class
 *
 * @since 0.10.0
 */
abstract class AbstractStyle
{
    /**
     * Style name
     *
     * @var string
     */
    protected $styleName;

    /**
     * Index number in Style collection for named style
     *
     * This number starts from one and defined in Style::setStyleValues()
     *
     * @var int|null
     */
    protected $index;

    /**
     * Aliases
     *
     * @var array
     */
    protected $aliases = array();

    /**
     * Is this an automatic style? (Used primarily in OpenDocument driver)
     *
     * @var bool
     * @since 0.11.0
     */
    private $isAuto = false;

    /**
     * Get style name
     *
     * @return string
     */
    public function getStyleName()
    {
        return $this->styleName;
    }

    /**
     * Set style name
     *
     * @param string $value
     * @return self
     */
    public function setStyleName($value)
    {
        $this->styleName = $value;

        return $this;
    }

    /**
     * Get index number
     *
     * @return int|null
     */
    public function getIndex()
    {
        return $this->index;
    }

    /**
     * Set index number
     *
     * @param int|null $value
     * @return self
     */
    public function setIndex($value = null)
    {
        $this->index = $this->setIntVal($value, $this->index);

        return $this;
    }

    /**
     * Get is automatic style flag
     *
     * @return bool
     */
    public function isAuto()
    {
        return $this->isAuto;
    }

    /**
     * Set is automatic style flag
     *
     * @param bool $value
     * @return self
     */
    public function setAuto($value = true)
    {
        $this->isAuto = $this->setBoolVal($value, $this->isAuto);

        return $this;
    }

    /**
     * Return style value of child style object, e.g. `left` from `Indentation` child style of `Paragraph`
     *
     * @param \PhpOffice\PhpWord\Style\AbstractStyle $substyleObject
     * @param string $substyleProperty
     * @return mixed
     * @since 0.12.0
     */
    public function getChildStyleValue($substyleObject, $substyleProperty)
    {
        if ($substyleObject !== null) {
            $method = "get{$substyleProperty}";

            return $substyleObject->$method();
        }

        return null;
    }

    /**
     * Set style value template method
     *
     * Some child classes have their own specific overrides.
     * Backward compability check for versions < 0.10.0 which use underscore
     * prefix for their private properties.
     * Check if the set method is exists. Throws an exception?
     *
     * @param string $key
     * @param string $value
     * @return self
     */
    public function setStyleValue($key, $value)
    {
        if (isset($this->aliases[$key])) {
            $key = $this->aliases[$key];
        }
        $method = 'set' . Text::removeUnderscorePrefix($key);
        if (method_exists($this, $method)) {
            $this->$method($value);
        }

        return $this;
    }

    /**
     * Set style by using associative array
     *
     * @param array $values
     * @return self
     */
    public function setStyleByArray($values = array())
    {
        foreach ($values as $key => $value) {
            $this->setStyleValue($key, $value);
        }

        return $this;
    }

    /**
     * Set default for null and empty value
     *
     * @param string $value (was: mixed)
     * @param string $default (was: mixed)
     * @return string (was: mixed)
     */
    protected function setNonEmptyVal($value, $default)
    {
        if ($value === null || $value == '') {
            $value = $default;
        }

        return $value;
    }

    /**
     * Set bool value
     *
     * @param bool $value
     * @param bool $default
     * @return bool
     */
    protected function setBoolVal($value, $default)
    {
        if (!is_bool($value)) {
            $value = $default;
        }

        return $value;
    }

    /**
     * Set numeric value
     *
     * @param mixed $value
     * @param int|float|null $default
     * @return int|float|null
     */
    protected function setNumericVal($value, $default = null)
    {
        if (!is_numeric($value)) {
            $value = $default;
        }

        return $value;
    }

    /**
     * Set integer value: Convert string that contains only numeric into integer
     *
     * @param int|null $value
     * @param int|null $default
     * @return int|null
     */
    protected function setIntVal($value, $default = null)
    {
        if (is_string($value) && (preg_match('/[^\d]/', $value) == 0)) {
            $value = (int) $value;
        }
        if (!is_numeric($value)) {
            $value = $default;
        } else {
            $value = (int) $value;
        }

        return $value;
    }

    /**
     * Set float value: Convert string that contains only numeric into float
     *
     * @param mixed $value
     * @param float|null $default
     * @return float|null
     */
    protected function setFloatVal($value, $default = null)
    {
        if (is_string($value) && (preg_match('/[^\d\.\,]/', $value) == 0)) {
            $value = (float) $value;
        }
        if (!is_numeric($value)) {
            $value = $default;
        }

        return $value;
    }

    /**
     * Set enum value
     *
     * @param mixed $value
     * @param array $enum
     * @param mixed $default
     *
     * @throws \InvalidArgumentException
     * @return mixed
     */
    protected function setEnumVal($value = null, $enum = array(), $default = null)
    {
        if ($value != null && trim($value) != '' && !empty($enum) && !in_array($value, $enum)) {
            throw new \InvalidArgumentException("Invalid style value: {$value} Options:" . implode(',', $enum));
        } elseif ($value === null || trim($value) == '') {
            $value = $default;
        }

        return $value;
    }

    /**
     * Set object value
     *
     * @param mixed $value
     * @param string $styleName
     * @param mixed &$style
     * @return mixed
     */
    protected function setObjectVal($value, $styleName, &$style)
    {
        $styleClass = substr(get_class($this), 0, strrpos(get_class($this), '\\')) . '\\' . $styleName;
        if (is_array($value)) {
            /** @var \PhpOffice\PhpWord\Style\AbstractStyle $style Type hint */
            if (!$style instanceof $styleClass) {
                $style = new $styleClass();
            }
            $style->setStyleByArray($value);
        } else {
            $style = $value;
        }

        return $style;
    }

    /**
     * Set $property value and set $pairProperty = false when $value = true
     *
     * @param bool &$property
     * @param bool &$pairProperty
     * @param bool $value
     * @return self
     */
    protected function setPairedVal(&$property, &$pairProperty, $value)
    {
        $property = $this->setBoolVal($value, $property);
        if ($value === true) {
            $pairProperty = false;
        }

        return $this;
    }

    /**
     * Set style using associative array
     *
     * @deprecated 0.11.0
     *
     * @param array $style
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setArrayStyle(array $style = array())
    {
        return $this->setStyleByArray($style);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Border.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Border style
 */
class Border extends AbstractStyle
{
    /**
     * Border Top Size
     *
     * @var int|float
     */
    protected $borderTopSize;

    /**
     * Border Top Color
     *
     * @var string
     */
    protected $borderTopColor;

    /**
     * Border Top Style
     *
     * @var string
     */
    protected $borderTopStyle;

    /**
     * Border Left Size
     *
     * @var int|float
     */
    protected $borderLeftSize;

    /**
     * Border Left Color
     *
     * @var string
     */
    protected $borderLeftColor;

    /**
     * Border Left Style
     *
     * @var string
     */
    protected $borderLeftStyle;

    /**
     * Border Right Size
     *
     * @var int|float
     */
    protected $borderRightSize;

    /**
     * Border Right Color
     *
     * @var string
     */
    protected $borderRightColor;

    /**
     * Border Right Style
     *
     * @var string
     */
    protected $borderRightStyle;

    /**
     * Border Bottom Size
     *
     * @var int|float
     */
    protected $borderBottomSize;

    /**
     * Border Bottom Color
     *
     * @var string
     */
    protected $borderBottomColor;

    /**
     * Border Bottom Style
     *
     * @var string
     */
    protected $borderBottomStyle;

    /**
     * Get border size
     *
     * @return int[]
     */
    public function getBorderSize()
    {
        return array(
            $this->getBorderTopSize(),
            $this->getBorderLeftSize(),
            $this->getBorderRightSize(),
            $this->getBorderBottomSize(),
        );
    }

    /**
     * Set border size
     *
     * @param int|float $value
     * @return self
     */
    public function setBorderSize($value = null)
    {
        $this->setBorderTopSize($value);
        $this->setBorderLeftSize($value);
        $this->setBorderRightSize($value);
        $this->setBorderBottomSize($value);

        return $this;
    }

    /**
     * Get border color
     *
     * @return string[]
     */
    public function getBorderColor()
    {
        return array(
            $this->getBorderTopColor(),
            $this->getBorderLeftColor(),
            $this->getBorderRightColor(),
            $this->getBorderBottomColor(),
        );
    }

    /**
     * Set border color
     *
     * @param string $value
     * @return self
     */
    public function setBorderColor($value = null)
    {
        $this->setBorderTopColor($value);
        $this->setBorderLeftColor($value);
        $this->setBorderRightColor($value);
        $this->setBorderBottomColor($value);

        return $this;
    }

    /**
     * Get border style
     *
     * @return string[]
     */
    public function getBorderStyle()
    {
        return array(
            $this->getBorderTopStyle(),
            $this->getBorderLeftStyle(),
            $this->getBorderRightStyle(),
            $this->getBorderBottomStyle(),
        );
    }

    /**
     * Set border style
     *
     * @param string $value
     * @return self
     */
    public function setBorderStyle($value = null)
    {
        $this->setBorderTopStyle($value);
        $this->setBorderLeftStyle($value);
        $this->setBorderRightStyle($value);
        $this->setBorderBottomStyle($value);

        return $this;
    }

    /**
     * Get border top size
     *
     * @return int|float
     */
    public function getBorderTopSize()
    {
        return $this->borderTopSize;
    }

    /**
     * Set border top size
     *
     * @param int|float $value
     * @return self
     */
    public function setBorderTopSize($value = null)
    {
        $this->borderTopSize = $this->setNumericVal($value, $this->borderTopSize);

        return $this;
    }

    /**
     * Get border top color
     *
     * @return string
     */
    public function getBorderTopColor()
    {
        return $this->borderTopColor;
    }

    /**
     * Set border top color
     *
     * @param string $value
     * @return self
     */
    public function setBorderTopColor($value = null)
    {
        $this->borderTopColor = $value;

        return $this;
    }

    /**
     * Get border top style
     *
     * @return string
     */
    public function getBorderTopStyle()
    {
        return $this->borderTopStyle;
    }

    /**
     * Set border top Style
     *
     * @param string $value
     * @return self
     */
    public function setBorderTopStyle($value = null)
    {
        $this->borderTopStyle = $value;

        return $this;
    }

    /**
     * Get border left size
     *
     * @return int|float
     */
    public function getBorderLeftSize()
    {
        return $this->borderLeftSize;
    }

    /**
     * Set border left size
     *
     * @param int|float $value
     * @return self
     */
    public function setBorderLeftSize($value = null)
    {
        $this->borderLeftSize = $this->setNumericVal($value, $this->borderLeftSize);

        return $this;
    }

    /**
     * Get border left color
     *
     * @return string
     */
    public function getBorderLeftColor()
    {
        return $this->borderLeftColor;
    }

    /**
     * Set border left color
     *
     * @param string $value
     * @return self
     */
    public function setBorderLeftColor($value = null)
    {
        $this->borderLeftColor = $value;

        return $this;
    }

    /**
     * Get border left style
     *
     * @return string
     */
    public function getBorderLeftStyle()
    {
        return $this->borderLeftStyle;
    }

    /**
     * Set border left style
     *
     * @param string $value
     * @return self
     */
    public function setBorderLeftStyle($value = null)
    {
        $this->borderLeftStyle = $value;

        return $this;
    }

    /**
     * Get border right size
     *
     * @return int|float
     */
    public function getBorderRightSize()
    {
        return $this->borderRightSize;
    }

    /**
     * Set border right size
     *
     * @param int|float $value
     * @return self
     */
    public function setBorderRightSize($value = null)
    {
        $this->borderRightSize = $this->setNumericVal($value, $this->borderRightSize);

        return $this;
    }

    /**
     * Get border right color
     *
     * @return string
     */
    public function getBorderRightColor()
    {
        return $this->borderRightColor;
    }

    /**
     * Set border right color
     *
     * @param string $value
     * @return self
     */
    public function setBorderRightColor($value = null)
    {
        $this->borderRightColor = $value;

        return $this;
    }

    /**
     * Get border right style
     *
     * @return string
     */
    public function getBorderRightStyle()
    {
        return $this->borderRightStyle;
    }

    /**
     * Set border right style
     *
     * @param string $value
     * @return self
     */
    public function setBorderRightStyle($value = null)
    {
        $this->borderRightStyle = $value;

        return $this;
    }

    /**
     * Get border bottom size
     *
     * @return int|float
     */
    public function getBorderBottomSize()
    {
        return $this->borderBottomSize;
    }

    /**
     * Set border bottom size
     *
     * @param int|float $value
     * @return self
     */
    public function setBorderBottomSize($value = null)
    {
        $this->borderBottomSize = $this->setNumericVal($value, $this->borderBottomSize);

        return $this;
    }

    /**
     * Get border bottom color
     *
     * @return string
     */
    public function getBorderBottomColor()
    {
        return $this->borderBottomColor;
    }

    /**
     * Set border bottom color
     *
     * @param string $value
     * @return self
     */
    public function setBorderBottomColor($value = null)
    {
        $this->borderBottomColor = $value;

        return $this;
    }

    /**
     * Get border bottom style
     *
     * @return string
     */
    public function getBorderBottomStyle()
    {
        return $this->borderBottomStyle;
    }

    /**
     * Set border bottom style
     *
     * @param string $value
     * @return self
     */
    public function setBorderBottomStyle($value = null)
    {
        $this->borderBottomStyle = $value;

        return $this;
    }

    /**
     * Check if any of the border is not null
     *
     * @return bool
     */
    public function hasBorder()
    {
        $borders = $this->getBorderSize();

        return $borders !== array_filter($borders, 'is_null');
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Cell.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\SimpleType\TblWidth;
use PhpOffice\PhpWord\SimpleType\VerticalJc;

/**
 * Table cell style
 */
class Cell extends Border
{
    /**
     * Vertical alignment constants
     *
     * @const string
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\VerticalJc::TOP instead
     */
    const VALIGN_TOP = 'top';
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\VerticalJc::CENTER instead
     */
    const VALIGN_CENTER = 'center';
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\VerticalJc::BOTTOM instead
     */
    const VALIGN_BOTTOM = 'bottom';
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\VerticalJc::BOTH instead
     */
    const VALIGN_BOTH = 'both';

    //Text direction constants
    /**
     * Left to Right, Top to Bottom
     */
    const TEXT_DIR_LRTB = 'lrTb';
    /**
     * Top to Bottom, Right to Left
     */
    const TEXT_DIR_TBRL = 'tbRl';
    /**
     * Bottom to Top, Left to Right
     */
    const TEXT_DIR_BTLR = 'btLr';
    /**
     * Left to Right, Top to Bottom Rotated
     */
    const TEXT_DIR_LRTBV = 'lrTbV';
    /**
     * Top to Bottom, Right to Left Rotated
     */
    const TEXT_DIR_TBRLV = 'tbRlV';
    /**
     * Top to Bottom, Left to Right Rotated
     */
    const TEXT_DIR_TBLRV = 'tbLrV';

    /**
     * Vertical merge (rowspan) constants
     *
     * @const string
     */
    const VMERGE_RESTART = 'restart';
    const VMERGE_CONTINUE = 'continue';

    /**
     * Default border color
     *
     * @const string
     */
    const DEFAULT_BORDER_COLOR = '000000';

    /**
     * Vertical align (top, center, both, bottom)
     *
     * @var string
     */
    private $vAlign;

    /**
     * Text Direction
     *
     * @var string
     */
    private $textDirection;

    /**
     * colspan
     *
     * @var int
     */
    private $gridSpan;

    /**
     * rowspan (restart, continue)
     *
     * - restart: Start/restart merged region
     * - continue: Continue merged region
     *
     * @var string
     */
    private $vMerge;

    /**
     * Shading
     *
     * @var \PhpOffice\PhpWord\Style\Shading
     */
    private $shading;

    /**
     * Width
     *
     * @var int
     */
    private $width;

    /**
     * Width unit
     *
     * @var string
     */
    private $unit = TblWidth::TWIP;

    /**
     * Get vertical align.
     *
     * @return string
     */
    public function getVAlign()
    {
        return $this->vAlign;
    }

    /**
     * Set vertical align
     *
     * @param string $value
     * @return self
     */
    public function setVAlign($value = null)
    {
        VerticalJc::validate($value);
        $this->vAlign = $this->setEnumVal($value, VerticalJc::values(), $this->vAlign);

        return $this;
    }

    /**
     * Get text direction.
     *
     * @return string
     */
    public function getTextDirection()
    {
        return $this->textDirection;
    }

    /**
     * Set text direction
     *
     * @param string $value
     * @return self
     */
    public function setTextDirection($value = null)
    {
        $enum = array(self::TEXT_DIR_BTLR, self::TEXT_DIR_TBRL);
        $this->textDirection = $this->setEnumVal($value, $enum, $this->textDirection);

        return $this;
    }

    /**
     * Get background
     *
     * @return string
     */
    public function getBgColor()
    {
        if ($this->shading !== null) {
            return $this->shading->getFill();
        }

        return null;
    }

    /**
     * Set background
     *
     * @param string $value
     * @return self
     */
    public function setBgColor($value = null)
    {
        return $this->setShading(array('fill' => $value));
    }

    /**
     * Get grid span (colspan).
     *
     * @return int
     */
    public function getGridSpan()
    {
        return $this->gridSpan;
    }

    /**
     * Set grid span (colspan)
     *
     * @param int $value
     * @return self
     */
    public function setGridSpan($value = null)
    {
        $this->gridSpan = $this->setIntVal($value, $this->gridSpan);

        return $this;
    }

    /**
     * Get vertical merge (rowspan).
     *
     * @return string
     */
    public function getVMerge()
    {
        return $this->vMerge;
    }

    /**
     * Set vertical merge (rowspan)
     *
     * @param string $value
     * @return self
     */
    public function setVMerge($value = null)
    {
        $enum = array(self::VMERGE_RESTART, self::VMERGE_CONTINUE);
        $this->vMerge = $this->setEnumVal($value, $enum, $this->vMerge);

        return $this;
    }

    /**
     * Get shading
     *
     * @return \PhpOffice\PhpWord\Style\Shading
     */
    public function getShading()
    {
        return $this->shading;
    }

    /**
     * Set shading
     *
     * @param mixed $value
     * @return self
     */
    public function setShading($value = null)
    {
        $this->setObjectVal($value, 'Shading', $this->shading);

        return $this;
    }

    /**
     * Get cell width
     *
     * @return int
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Set cell width
     *
     * @param int $value
     * @return self
     */
    public function setWidth($value)
    {
        $this->setIntVal($value);

        return $this;
    }

    /**
     * Get width unit
     *
     * @return string
     */
    public function getUnit()
    {
        return $this->unit;
    }

    /**
     * Set width unit
     *
     * @param string $value
     */
    public function setUnit($value)
    {
        $this->unit = $this->setEnumVal($value, array(TblWidth::AUTO, TblWidth::PERCENT, TblWidth::TWIP), TblWidth::TWIP);

        return $this;
    }

    /**
     * Get default border color
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getDefaultBorderColor()
    {
        return self::DEFAULT_BORDER_COLOR;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Chart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Chart style
 *
 * @since 0.12.0
 */
class Chart extends AbstractStyle
{
    /**
     * Width (in EMU)
     *
     * @var int
     */
    private $width = 1000000;

    /**
     * Height (in EMU)
     *
     * @var int
     */
    private $height = 1000000;

    /**
     * Is 3D; applies to pie, bar, line, area
     *
     * @var bool
     */
    private $is3d = false;

    /**
     * A list of colors to use in the chart
     *
     * @var array
     */
    private $colors = array();

    /**
     * Chart title
     *
     * @var string
     */
    private $title = null;

    /**
     * Chart legend visibility
     *
     * @var bool
     */
    private $showLegend = false;

    /**
     * Chart legend Position.
     * Possible values are 'r', 't', 'b', 'l', 'tr'
     *
     * @var string
     */
    private $legendPosition = 'r';

    /**
     * A list of display options for data labels
     *
     * @var array
     */
    private $dataLabelOptions = array(
        'showVal'          => true, // value
        'showCatName'      => true, // category name
        'showLegendKey'    => false, //show the cart legend
        'showSerName'      => false, // series name
        'showPercent'      => false,
        'showLeaderLines'  => false,
        'showBubbleSize'   => false,
    );

    /**
     * A string that tells the writer where to write chart labels or to skip
     * "nextTo" - sets labels next to the axis (bar graphs on the left) (default)
     * "low" - labels on the left side of the graph
     * "high" - labels on the right side of the graph
     *
     * @var string
     */
    private $categoryLabelPosition = 'nextTo';

    /**
     * A string that tells the writer where to write chart labels or to skip
     * "nextTo" - sets labels next to the axis (bar graphs on the bottom) (default)
     * "low" - labels are below the graph
     * "high" - labels above the graph
     *
     * @var string
     */
    private $valueLabelPosition = 'nextTo';

    /**
     * @var string
     */
    private $categoryAxisTitle;

    /**
     * @var string
     */
    private $valueAxisTitle;

    /**
     * The position for major tick marks
     * Possible values are 'in', 'out', 'cross', 'none'
     *
     * @var string
     */
    private $majorTickMarkPos = 'none';

    /**
     * Show labels for axis
     *
     * @var bool
     */
    private $showAxisLabels = false;

    /**
     * Show Gridlines for Y-Axis
     *
     * @var bool
     */
    private $gridY = false;

    /**
     * Show Gridlines for X-Axis
     *
     * @var bool
     */
    private $gridX = false;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get width
     *
     * @return int
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Set width
     *
     * @param int $value
     * @return self
     */
    public function setWidth($value = null)
    {
        $this->width = $this->setIntVal($value, $this->width);

        return $this;
    }

    /**
     * Get height
     *
     * @return int
     */
    public function getHeight()
    {
        return $this->height;
    }

    /**
     * Set height
     *
     * @param int $value
     * @return self
     */
    public function setHeight($value = null)
    {
        $this->height = $this->setIntVal($value, $this->height);

        return $this;
    }

    /**
     * Is 3D
     *
     * @return bool
     */
    public function is3d()
    {
        return $this->is3d;
    }

    /**
     * Set 3D
     *
     * @param bool $value
     * @return self
     */
    public function set3d($value = true)
    {
        $this->is3d = $this->setBoolVal($value, $this->is3d);

        return $this;
    }

    /**
     * Get the list of colors to use in a chart.
     *
     * @return array
     */
    public function getColors()
    {
        return $this->colors;
    }

    /**
     * Set the colors to use in a chart.
     *
     * @param array $value a list of colors to use in the chart
     * @return self
     */
    public function setColors($value = array())
    {
        $this->colors = $value;

        return $this;
    }

    /**
     * Get the chart title
     *
     * @return string
     */
    public function getTitle()
    {
        return $this->title;
    }

    /**
     * Set the chart title
     *
     * @param string $value
     * @return self
     */
    public function setTitle($value = null)
    {
        $this->title = $value;

        return $this;
    }

    /**
     * Get chart legend visibility
     *
     * @return bool
     */
    public function isShowLegend()
    {
        return $this->showLegend;
    }

    /**
     * Set chart legend visibility
     *
     * @param bool $value
     * @return self
     */
    public function setShowLegend($value = false)
    {
        $this->showLegend = $value;

        return $this;
    }

    /**
     * Get chart legend position
     *
     * @return string
     */
    public function getLegendPosition()
    {
        return $this->legendPosition;
    }

    /**
     * Set chart legend position. choices:
     * "r" - right of chart
     * "b" - bottom of chart
     * "t" - top of chart
     * "l" - left of chart
     * "tr" - top right of chart
     *
     * default: right
     *
     * @param string $legendPosition
     * @return self
     */
    public function setLegendPosition($legendPosition = 'r')
    {
        $enum = array('r', 'b', 't', 'l', 'tr');
        $this->legendPosition = $this->setEnumVal($legendPosition, $enum, $this->legendPosition);

        return $this;
    }

    /*
     * Show labels for axis
     *
     * @return bool
     */
    public function showAxisLabels()
    {
        return $this->showAxisLabels;
    }

    /**
     * Set show Gridlines for Y-Axis
     *
     * @param bool $value
     * @return self
     */
    public function setShowAxisLabels($value = true)
    {
        $this->showAxisLabels = $this->setBoolVal($value, $this->showAxisLabels);

        return $this;
    }

    /**
     * get the list of options for data labels
     *
     * @return array
     */
    public function getDataLabelOptions()
    {
        return $this->dataLabelOptions;
    }

    /**
     * Set values for data label options.
     * This will only change values for options defined in $this->dataLabelOptions, and cannot create new ones.
     *
     * @param array $values [description]
     */
    public function setDataLabelOptions($values = array())
    {
        foreach (array_keys($this->dataLabelOptions) as $option) {
            if (isset($values[$option])) {
                $this->dataLabelOptions[$option] = $this->setBoolVal(
                    $values[$option],
                    $this->dataLabelOptions[$option]
                );
            }
        }
    }

    /*
     * Show Gridlines for Y-Axis
     *
     * @return bool
     */
    public function showGridY()
    {
        return $this->gridY;
    }

    /**
     * Set show Gridlines for Y-Axis
     *
     * @param bool $value
     * @return self
     */
    public function setShowGridY($value = true)
    {
        $this->gridY = $this->setBoolVal($value, $this->gridY);

        return $this;
    }

    /**
     * Get the categoryLabelPosition setting
     *
     * @return string
     */
    public function getCategoryLabelPosition()
    {
        return $this->categoryLabelPosition;
    }

    /**
     * Set the categoryLabelPosition setting
     * "none" - skips writing  labels
     * "nextTo" - sets labels next to the  (bar graphs on the left)
     * "low" - labels on the left side of the graph
     * "high" - labels on the right side of the graph
     *
     * @param mixed $labelPosition
     * @return self
     */
    public function setCategoryLabelPosition($labelPosition)
    {
        $enum = array('nextTo', 'low', 'high');
        $this->categoryLabelPosition = $this->setEnumVal($labelPosition, $enum, $this->categoryLabelPosition);

        return $this;
    }

    /**
     * Get the valueAxisLabelPosition setting
     *
     * @return string
     */
    public function getValueLabelPosition()
    {
        return $this->valueLabelPosition;
    }

    /**
     * Set the valueLabelPosition setting
     * "none" - skips writing labels
     * "nextTo" - sets labels next to the value
     * "low" - sets labels are below the graph
     * "high" - sets labels above the graph
     *
     * @param string
     * @param mixed $labelPosition
     */
    public function setValueLabelPosition($labelPosition)
    {
        $enum = array('nextTo', 'low', 'high');
        $this->valueLabelPosition = $this->setEnumVal($labelPosition, $enum, $this->valueLabelPosition);

        return $this;
    }

    /**
     * Get the categoryAxisTitle
     * @return string
     */
    public function getCategoryAxisTitle()
    {
        return $this->categoryAxisTitle;
    }

    /**
     * Set the title that appears on the category side of the chart
     * @param string $axisTitle
     */
    public function setCategoryAxisTitle($axisTitle)
    {
        $this->categoryAxisTitle = $axisTitle;

        return $this;
    }

    /**
     * Get the valueAxisTitle
     * @return string
     */
    public function getValueAxisTitle()
    {
        return $this->valueAxisTitle;
    }

    /**
     * Set the title that appears on the value side of the chart
     * @param string $axisTitle
     */
    public function setValueAxisTitle($axisTitle)
    {
        $this->valueAxisTitle = $axisTitle;

        return $this;
    }

    public function getMajorTickPosition()
    {
        return $this->majorTickMarkPos;
    }

    /**
     * Set the position for major tick marks
     * @param string $position
     */
    public function setMajorTickPosition($position)
    {
        $enum = array('in', 'out', 'cross', 'none');
        $this->majorTickMarkPos = $this->setEnumVal($position, $enum, $this->majorTickMarkPos);
    }

    /**
     * Show Gridlines for X-Axis
     *
     * @return bool
     */
    public function showGridX()
    {
        return $this->gridX;
    }

    /**
     * Set show Gridlines for X-Axis
     *
     * @param bool $value
     * @return self
     */
    public function setShowGridX($value = true)
    {
        $this->gridX = $this->setBoolVal($value, $this->gridX);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Extrusion.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * 3D extrusion style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-o_CT_Extrusion.html
 * @since 0.12.0
 */
class Extrusion extends AbstractStyle
{
    /**
     * Type constants
     *
     * @const string
     */
    const EXTRUSION_PARALLEL = 'parallel';
    const EXTRUSION_PERSPECTIVE = 'perspective';

    /**
     * Type: parallel|perspective
     *
     * @var string
     */
    private $type;

    /**
     * Color
     *
     * @var string
     */
    private $color;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setType($value = null)
    {
        $enum = array(self::EXTRUSION_PARALLEL, self::EXTRUSION_PERSPECTIVE);
        $this->type = $this->setEnumVal($value, $enum, null);

        return $this;
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Fill.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Fill style
 *
 * There are still lot of interesting things for this style that can be added, including gradient. See @see .
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-v_CT_Fill.html
 * @since 0.12.0
 */
class Fill extends AbstractStyle
{
    /**
     * Color
     *
     * @var string
     */
    private $color;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Font.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Font style
 */
class Font extends AbstractStyle
{
    /**
     * Underline types
     *
     * @const string
     */
    const UNDERLINE_NONE = 'none';
    const UNDERLINE_DASH = 'dash';
    const UNDERLINE_DASHHEAVY = 'dashHeavy';
    const UNDERLINE_DASHLONG = 'dashLong';
    const UNDERLINE_DASHLONGHEAVY = 'dashLongHeavy';
    const UNDERLINE_DOUBLE = 'dbl';
    /**
     * @deprecated use UNDERLINE_DOTHASH instead, TODO remove in version 1.0
     */
    const UNDERLINE_DOTHASH = 'dotDash';  // Incorrect spelling, for backwards compatibility
    /**
     * @deprecated use UNDERLINE_DOTDASHHEAVY instead, TODO remove in version 1.0
     */
    const UNDERLINE_DOTHASHHEAVY = 'dotDashHeavy';  // Incorrect spelling, for backwards compatibility
    const UNDERLINE_DOTDASH = 'dotDash';
    const UNDERLINE_DOTDASHHEAVY = 'dotDashHeavy';
    const UNDERLINE_DOTDOTDASH = 'dotDotDash';
    const UNDERLINE_DOTDOTDASHHEAVY = 'dotDotDashHeavy';
    const UNDERLINE_DOTTED = 'dotted';
    const UNDERLINE_DOTTEDHEAVY = 'dottedHeavy';
    const UNDERLINE_HEAVY = 'heavy';
    const UNDERLINE_SINGLE = 'single';
    const UNDERLINE_WAVY = 'wavy';
    const UNDERLINE_WAVYDOUBLE = 'wavyDbl';
    const UNDERLINE_WAVYHEAVY = 'wavyHeavy';
    const UNDERLINE_WORDS = 'words';

    /**
     * Foreground colors
     *
     * @const string
     */
    const FGCOLOR_YELLOW = 'yellow';
    const FGCOLOR_LIGHTGREEN = 'green';
    const FGCOLOR_CYAN = 'cyan';
    const FGCOLOR_MAGENTA = 'magenta';
    const FGCOLOR_BLUE = 'blue';
    const FGCOLOR_RED = 'red';
    const FGCOLOR_DARKBLUE = 'darkBlue';
    const FGCOLOR_DARKCYAN = 'darkCyan';
    const FGCOLOR_DARKGREEN = 'darkGreen';
    const FGCOLOR_DARKMAGENTA = 'darkMagenta';
    const FGCOLOR_DARKRED = 'darkRed';
    const FGCOLOR_DARKYELLOW = 'darkYellow';
    const FGCOLOR_DARKGRAY = 'darkGray';
    const FGCOLOR_LIGHTGRAY = 'lightGray';
    const FGCOLOR_BLACK = 'black';

    /**
     * Aliases
     *
     * @var array
     */
    protected $aliases = array('line-height' => 'lineHeight', 'letter-spacing' => 'spacing');

    /**
     * Font style type
     *
     * @var string
     */
    private $type;

    /**
     * Font name
     *
     * @var string
     */
    private $name;

    /**
     * Font Content Type
     *
     * @var string
     */
    private $hint;

    /**
     * Font size
     *
     * @var int|float
     */
    private $size;

    /**
     * Font color
     *
     * @var string
     */
    private $color;

    /**
     * Bold
     *
     * @var bool
     */
    private $bold;

    /**
     * Italic
     *
     * @var bool
     */
    private $italic;

    /**
     * Undeline
     *
     * @var string
     */
    private $underline = self::UNDERLINE_NONE;

    /**
     * Superscript
     *
     * @var bool
     */
    private $superScript = false;

    /**
     * Subscript
     *
     * @var bool
     */
    private $subScript = false;

    /**
     * Strikethrough
     *
     * @var bool
     */
    private $strikethrough;

    /**
     * Double strikethrough
     *
     * @var bool
     */
    private $doubleStrikethrough;

    /**
     * Small caps
     *
     * @var bool
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_smallCaps-1.html
     */
    private $smallCaps;

    /**
     * All caps
     *
     * @var bool
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_caps-1.html
     */
    private $allCaps;

    /**
     * Foreground/highlight
     *
     * @var string
     */
    private $fgColor;

    /**
     * Expanded/compressed text: 0-600 (percent)
     *
     * @var int
     * @since 0.12.0
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_w-1.html
     */
    private $scale;

    /**
     * Character spacing adjustment: twip
     *
     * @var int|float
     * @since 0.12.0
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_spacing-2.html
     */
    private $spacing;

    /**
     * Font kerning: halfpoint
     *
     * @var int|float
     * @since 0.12.0
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_kern-1.html
     */
    private $kerning;

    /**
     * Paragraph style
     *
     * @var \PhpOffice\PhpWord\Style\Paragraph
     */
    private $paragraph;

    /**
     * Shading
     *
     * @var \PhpOffice\PhpWord\Style\Shading
     */
    private $shading;

    /**
     * Right to left languages
     *
     * @var bool
     */
    private $rtl;

    /**
     * noProof (disables AutoCorrect)
     *
     * @var bool
     * http://www.datypic.com/sc/ooxml/e-w_noProof-1.html
     */
    private $noProof;

    /**
     * Languages
     *
     * @var \PhpOffice\PhpWord\Style\Language
     */
    private $lang;

    /**
     * Hidden text
     *
     * @var bool
     * @see  http://www.datypic.com/sc/ooxml/e-w_vanish-1.html
     */
    private $hidden;

    /**
     * Vertically Raised or Lowered Text
     *
     * @var int Signed Half-Point Measurement
     * @see http://www.datypic.com/sc/ooxml/e-w_position-1.html
     */
    private $position;

    /**
     * Create new font style
     *
     * @param string $type Type of font
     * @param array|string|\PhpOffice\PhpWord\Style\AbstractStyle $paragraph Paragraph styles definition
     */
    public function __construct($type = 'text', $paragraph = null)
    {
        $this->type = $type;
        $this->setParagraph($paragraph);
    }

    /**
     * Get style values
     *
     * @return array
     * @since 0.12.0
     */
    public function getStyleValues()
    {
        $styles = array(
            'name'          => $this->getStyleName(),
            'basic'         => array(
                'name'      => $this->getName(),
                'size'      => $this->getSize(),
                'color'     => $this->getColor(),
                'hint'      => $this->getHint(),
            ),
            'style'         => array(
                'bold'      => $this->isBold(),
                'italic'    => $this->isItalic(),
                'underline' => $this->getUnderline(),
                'strike'    => $this->isStrikethrough(),
                'dStrike'   => $this->isDoubleStrikethrough(),
                'super'     => $this->isSuperScript(),
                'sub'       => $this->isSubScript(),
                'smallCaps' => $this->isSmallCaps(),
                'allCaps'   => $this->isAllCaps(),
                'fgColor'   => $this->getFgColor(),
                'hidden'    => $this->isHidden(),
            ),
            'spacing'       => array(
                'scale'     => $this->getScale(),
                'spacing'   => $this->getSpacing(),
                'kerning'   => $this->getKerning(),
                'position'  => $this->getPosition(),
            ),
            'paragraph'     => $this->getParagraph(),
            'rtl'           => $this->isRTL(),
            'shading'       => $this->getShading(),
            'lang'          => $this->getLang(),
        );

        return $styles;
    }

    /**
     * Get style type
     *
     * @return string
     */
    public function getStyleType()
    {
        return $this->type;
    }

    /**
     * Get font name
     *
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set font name
     *
     * @param string $value
     * @return self
     */
    public function setName($value = null)
    {
        $this->name = $value;

        return $this;
    }

    /**
     * Get Font Content Type
     *
     * @return string
     */
    public function getHint()
    {
        return $this->hint;
    }

    /**
     * Set Font Content Type
     *
     * @param string $value
     * @return self
     */
    public function setHint($value = null)
    {
        $this->hint = $value;

        return $this;
    }

    /**
     * Get font size
     *
     * @return int|float
     */
    public function getSize()
    {
        return $this->size;
    }

    /**
     * Set font size
     *
     * @param int|float $value
     * @return self
     */
    public function setSize($value = null)
    {
        $this->size = $this->setNumericVal($value, $this->size);

        return $this;
    }

    /**
     * Get font color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set font color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }

    /**
     * Get bold
     *
     * @return bool
     */
    public function isBold()
    {
        return $this->bold;
    }

    /**
     * Set bold
     *
     * @param bool $value
     * @return self
     */
    public function setBold($value = true)
    {
        $this->bold = $this->setBoolVal($value, $this->bold);

        return $this;
    }

    /**
     * Get italic
     *
     * @return bool
     */
    public function isItalic()
    {
        return $this->italic;
    }

    /**
     * Set italic
     *
     * @param bool $value
     * @return self
     */
    public function setItalic($value = true)
    {
        $this->italic = $this->setBoolVal($value, $this->italic);

        return $this;
    }

    /**
     * Get underline
     *
     * @return string
     */
    public function getUnderline()
    {
        return $this->underline;
    }

    /**
     * Set underline
     *
     * @param string $value
     * @return self
     */
    public function setUnderline($value = self::UNDERLINE_NONE)
    {
        $this->underline = $this->setNonEmptyVal($value, self::UNDERLINE_NONE);

        return $this;
    }

    /**
     * Get superscript
     *
     * @return bool
     */
    public function isSuperScript()
    {
        return $this->superScript;
    }

    /**
     * Set superscript
     *
     * @param bool $value
     * @return self
     */
    public function setSuperScript($value = true)
    {
        return $this->setPairedVal($this->superScript, $this->subScript, $value);
    }

    /**
     * Get subscript
     *
     * @return bool
     */
    public function isSubScript()
    {
        return $this->subScript;
    }

    /**
     * Set subscript
     *
     * @param bool $value
     * @return self
     */
    public function setSubScript($value = true)
    {
        return $this->setPairedVal($this->subScript, $this->superScript, $value);
    }

    /**
     * Get strikethrough
     *
     * @return bool
     */
    public function isStrikethrough()
    {
        return $this->strikethrough;
    }

    /**
     * Set strikethrough
     *
     * @param bool $value
     * @return self
     */
    public function setStrikethrough($value = true)
    {
        return $this->setPairedVal($this->strikethrough, $this->doubleStrikethrough, $value);
    }

    /**
     * Get double strikethrough
     *
     * @return bool
     */
    public function isDoubleStrikethrough()
    {
        return $this->doubleStrikethrough;
    }

    /**
     * Set double strikethrough
     *
     * @param bool $value
     * @return self
     */
    public function setDoubleStrikethrough($value = true)
    {
        return $this->setPairedVal($this->doubleStrikethrough, $this->strikethrough, $value);
    }

    /**
     * Get small caps
     *
     * @return bool
     */
    public function isSmallCaps()
    {
        return $this->smallCaps;
    }

    /**
     * Set small caps
     *
     * @param bool $value
     * @return self
     */
    public function setSmallCaps($value = true)
    {
        return $this->setPairedVal($this->smallCaps, $this->allCaps, $value);
    }

    /**
     * Get all caps
     *
     * @return bool
     */
    public function isAllCaps()
    {
        return $this->allCaps;
    }

    /**
     * Set all caps
     *
     * @param bool $value
     * @return self
     */
    public function setAllCaps($value = true)
    {
        return $this->setPairedVal($this->allCaps, $this->smallCaps, $value);
    }

    /**
     * Get foreground/highlight color
     *
     * @return string
     */
    public function getFgColor()
    {
        return $this->fgColor;
    }

    /**
     * Set foreground/highlight color
     *
     * @param string $value
     * @return self
     */
    public function setFgColor($value = null)
    {
        $this->fgColor = $value;

        return $this;
    }

    /**
     * Get background
     *
     * @return string
     */
    public function getBgColor()
    {
        return $this->getChildStyleValue($this->shading, 'fill');
    }

    /**
     * Set background
     *
     * @param string $value
     * @return \PhpOffice\PhpWord\Style\Table
     */
    public function setBgColor($value = null)
    {
        $this->setShading(array('fill' => $value));
    }

    /**
     * Get scale
     *
     * @return int
     */
    public function getScale()
    {
        return $this->scale;
    }

    /**
     * Set scale
     *
     * @param int $value
     * @return self
     */
    public function setScale($value = null)
    {
        $this->scale = $this->setIntVal($value, null);

        return $this;
    }

    /**
     * Get font spacing
     *
     * @return int|float
     */
    public function getSpacing()
    {
        return $this->spacing;
    }

    /**
     * Set font spacing
     *
     * @param int|float $value
     * @return self
     */
    public function setSpacing($value = null)
    {
        $this->spacing = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get font kerning
     *
     * @return int|float
     */
    public function getKerning()
    {
        return $this->kerning;
    }

    /**
     * Set font kerning
     *
     * @param int|float $value
     * @return self
     */
    public function setKerning($value = null)
    {
        $this->kerning = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get noProof (disables autocorrect)
     *
     * @return bool
     */
    public function isNoProof()
    {
        return $this->noProof;
    }

    /**
     * Set noProof (disables autocorrect)
     *
     * @param bool $value
     * @return $this
     */
    public function setNoProof($value = false)
    {
        $this->noProof = $value;

        return $this;
    }

    /**
     * Get line height
     *
     * @return int|float
     */
    public function getLineHeight()
    {
        return $this->getParagraph()->getLineHeight();
    }

    /**
     * Set lineheight
     *
     * @param int|float|string $value
     * @return self
     */
    public function setLineHeight($value)
    {
        $this->setParagraph(array('lineHeight' => $value));

        return $this;
    }

    /**
     * Get paragraph style
     *
     * @return \PhpOffice\PhpWord\Style\Paragraph
     */
    public function getParagraph()
    {
        return $this->paragraph;
    }

    /**
     * Set Paragraph
     *
     * @param mixed $value
     * @return self
     */
    public function setParagraph($value = null)
    {
        $this->setObjectVal($value, 'Paragraph', $this->paragraph);

        return $this;
    }

    /**
     * Get rtl
     *
     * @return bool
     */
    public function isRTL()
    {
        return $this->rtl;
    }

    /**
     * Set rtl
     *
     * @param bool $value
     * @return self
     */
    public function setRTL($value = true)
    {
        $this->rtl = $this->setBoolVal($value, $this->rtl);

        return $this;
    }

    /**
     * Get shading
     *
     * @return \PhpOffice\PhpWord\Style\Shading
     */
    public function getShading()
    {
        return $this->shading;
    }

    /**
     * Set shading
     *
     * @param mixed $value
     * @return self
     */
    public function setShading($value = null)
    {
        $this->setObjectVal($value, 'Shading', $this->shading);

        return $this;
    }

    /**
     * Get language
     *
     * @return \PhpOffice\PhpWord\Style\Language
     */
    public function getLang()
    {
        return $this->lang;
    }

    /**
     * Set language
     *
     * @param mixed $value
     * @return self
     */
    public function setLang($value = null)
    {
        if (is_string($value) && $value != '') {
            $value = new Language($value);
        }
        $this->setObjectVal($value, 'Language', $this->lang);

        return $this;
    }

    /**
     * Get bold
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getBold()
    {
        return $this->isBold();
    }

    /**
     * Get italic
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getItalic()
    {
        return $this->isItalic();
    }

    /**
     * Get superscript
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getSuperScript()
    {
        return $this->isSuperScript();
    }

    /**
     * Get subscript
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getSubScript()
    {
        return $this->isSubScript();
    }

    /**
     * Get strikethrough
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getStrikethrough()
    {
        return $this->isStrikethrough();
    }

    /**
     * Get paragraph style
     *
     * @deprecated 0.11.0
     *
     * @codeCoverageIgnore
     */
    public function getParagraphStyle()
    {
        return $this->getParagraph();
    }

    /**
     * Get hidden text
     *
     * @return bool
     */
    public function isHidden()
    {
        return $this->hidden;
    }

    /**
     * Set hidden text
     *
     * @param bool $value
     * @return self
     */
    public function setHidden($value = true)
    {
        $this->hidden = $this->setBoolVal($value, $this->hidden);

        return $this;
    }

    /**
     * Get position
     *
     * @return int
     */
    public function getPosition()
    {
        return $this->position;
    }

    /**
     * Set position
     *
     * @param int $value
     * @return self
     */
    public function setPosition($value = null)
    {
        $this->position = $this->setIntVal($value, null);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Frame.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\SimpleType\Jc;

/**
 * Frame defines the size and position of an object
 *
 * Width, height, left/hpos, top/vpos, hrel, vrel, wrap, zindex
 *
 * @since 0.12.0
 * @todo Make existing style (image, textbox, etc) use this style
 */
class Frame extends AbstractStyle
{
    /**
     * Length unit
     *
     * @const string
     */
    const UNIT_PT = 'pt'; // Mostly for shapes
    const UNIT_PX = 'px'; // Mostly for images

    /**
     * General positioning options.
     *
     * @const string
     */
    const POS_ABSOLUTE = 'absolute';
    const POS_RELATIVE = 'relative';

    /**
     * Horizontal/vertical value
     *
     * @const string
     */
    const POS_CENTER = 'center';
    const POS_LEFT = 'left';
    const POS_RIGHT = 'right';
    const POS_TOP = 'top';
    const POS_BOTTOM = 'bottom';
    const POS_INSIDE = 'inside';
    const POS_OUTSIDE = 'outside';

    /**
     * Position relative to
     *
     * @const string
     */
    const POS_RELTO_MARGIN = 'margin';
    const POS_RELTO_PAGE = 'page';
    const POS_RELTO_COLUMN = 'column'; // horizontal only
    const POS_RELTO_CHAR = 'char'; // horizontal only
    const POS_RELTO_TEXT = 'text'; // vertical only
    const POS_RELTO_LINE = 'line'; // vertical only
    const POS_RELTO_LMARGIN = 'left-margin-area'; // horizontal only
    const POS_RELTO_RMARGIN = 'right-margin-area'; // horizontal only
    const POS_RELTO_TMARGIN = 'top-margin-area'; // vertical only
    const POS_RELTO_BMARGIN = 'bottom-margin-area'; // vertical only
    const POS_RELTO_IMARGIN = 'inner-margin-area';
    const POS_RELTO_OMARGIN = 'outer-margin-area';

    /**
     * Wrap type
     *
     * @const string
     */
    const WRAP_INLINE = 'inline';
    const WRAP_SQUARE = 'square';
    const WRAP_TIGHT = 'tight';
    const WRAP_THROUGH = 'through';
    const WRAP_TOPBOTTOM = 'topAndBottom';
    const WRAP_BEHIND = 'behind';
    const WRAP_INFRONT = 'infront';

    /**
     * @var string
     */
    private $alignment = '';

    /**
     * Unit
     *
     * @var string
     */
    private $unit = 'pt';

    /**
     * Width
     *
     * @var int|float
     */
    private $width;

    /**
     * Height
     *
     * @var int|float
     */
    private $height;

    /**
     * Leftmost (horizontal) position
     *
     * @var int|float
     */
    private $left = 0;

    /**
     * Topmost (vertical) position
     *
     * @var int|float
     */
    private $top = 0;

    /**
     * Position type: absolute|relative
     *
     * @var string
     */
    private $pos;

    /**
     * Horizontal position
     *
     * @var string
     */
    private $hPos;

    /**
     * Horizontal position relative to
     *
     * @var string
     */
    private $hPosRelTo;

    /**
     * Vertical position
     *
     * @var string
     */
    private $vPos;

    /**
     * Vertical position relative to
     *
     * @var string
     */
    private $vPosRelTo;

    /**
     * Wrap type
     *
     * @var string
     */
    private $wrap;

    /**
     * Top wrap distance
     *
     * @var float
     */
    private $wrapDistanceTop;

    /**
     * Bottom wrap distance
     *
     * @var float
     */
    private $wrapDistanceBottom;

    /**
     * Left wrap distance
     *
     * @var float
     */
    private $wrapDistanceLeft;

    /**
     * Right wrap distance
     *
     * @var float
     */
    private $wrapDistanceRight;

    /**
     * Vertically raised or lowered text
     *
     * @var int
     * @see http://www.datypic.com/sc/ooxml/e-w_position-1.html
     */
    private $position;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    public function getAlignment()
    {
        return $this->alignment;
    }

    /**
     * @since 0.13.0
     *
     * @param string $value
     *
     * @return self
     */
    public function setAlignment($value)
    {
        if (Jc::isValid($value)) {
            $this->alignment = $value;
        }

        return $this;
    }

    /**
     * @deprecated 0.13.0 Use the `getAlignment` method instead.
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getAlign()
    {
        return $this->getAlignment();
    }

    /**
     * @deprecated 0.13.0 Use the `setAlignment` method instead.
     *
     * @param string $value
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setAlign($value = null)
    {
        return $this->setAlignment($value);
    }

    /**
     * Get unit
     *
     * @return string
     */
    public function getUnit()
    {
        return $this->unit;
    }

    /**
     * Set unit
     *
     * @param string $value
     * @return self
     */
    public function setUnit($value)
    {
        $this->unit = $value;

        return $this;
    }

    /**
     * Get width
     *
     * @return int|float
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Set width
     *
     * @param int|float $value
     * @return self
     */
    public function setWidth($value = null)
    {
        $this->width = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get height
     *
     * @return int|float
     */
    public function getHeight()
    {
        return $this->height;
    }

    /**
     * Set height
     *
     * @param int|float $value
     * @return self
     */
    public function setHeight($value = null)
    {
        $this->height = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get left
     *
     * @return int|float
     */
    public function getLeft()
    {
        return $this->left;
    }

    /**
     * Set left
     *
     * @param int|float $value
     * @return self
     */
    public function setLeft($value = 0)
    {
        $this->left = $this->setNumericVal($value, 0);

        return $this;
    }

    /**
     * Get topmost position
     *
     * @return int|float
     */
    public function getTop()
    {
        return $this->top;
    }

    /**
     * Set topmost position
     *
     * @param int|float $value
     * @return self
     */
    public function setTop($value = 0)
    {
        $this->top = $this->setNumericVal($value, 0);

        return $this;
    }

    /**
     * Get position type
     *
     * @return string
     */
    public function getPos()
    {
        return $this->pos;
    }

    /**
     * Set position type
     *
     * @param string $value
     * @return self
     */
    public function setPos($value)
    {
        $enum = array(
            self::POS_ABSOLUTE,
            self::POS_RELATIVE,
        );
        $this->pos = $this->setEnumVal($value, $enum, $this->pos);

        return $this;
    }

    /**
     * Get horizontal position
     *
     * @return string
     */
    public function getHPos()
    {
        return $this->hPos;
    }

    /**
     * Set horizontal position
     *
     * @since 0.12.0 "absolute" option is available.
     *
     * @param string $value
     * @return self
     */
    public function setHPos($value)
    {
        $enum = array(
            self::POS_ABSOLUTE,
            self::POS_LEFT,
            self::POS_CENTER,
            self::POS_RIGHT,
            self::POS_INSIDE,
            self::POS_OUTSIDE,
        );
        $this->hPos = $this->setEnumVal($value, $enum, $this->hPos);

        return $this;
    }

    /**
     * Get vertical position
     *
     * @return string
     */
    public function getVPos()
    {
        return $this->vPos;
    }

    /**
     * Set vertical position
     *
     * @since 0.12.0 "absolute" option is available.
     *
     * @param string $value
     * @return self
     */
    public function setVPos($value)
    {
        $enum = array(
            self::POS_ABSOLUTE,
            self::POS_TOP,
            self::POS_CENTER,
            self::POS_BOTTOM,
            self::POS_INSIDE,
            self::POS_OUTSIDE,
        );
        $this->vPos = $this->setEnumVal($value, $enum, $this->vPos);

        return $this;
    }

    /**
     * Get horizontal position relative to
     *
     * @return string
     */
    public function getHPosRelTo()
    {
        return $this->hPosRelTo;
    }

    /**
     * Set horizontal position relative to
     *
     * @param string $value
     * @return self
     */
    public function setHPosRelTo($value)
    {
        $enum = array(
            self::POS_RELTO_MARGIN,
            self::POS_RELTO_PAGE,
            self::POS_RELTO_COLUMN,
            self::POS_RELTO_CHAR,
            self::POS_RELTO_LMARGIN,
            self::POS_RELTO_RMARGIN,
            self::POS_RELTO_IMARGIN,
            self::POS_RELTO_OMARGIN,
        );
        $this->hPosRelTo = $this->setEnumVal($value, $enum, $this->hPosRelTo);

        return $this;
    }

    /**
     * Get vertical position relative to
     *
     * @return string
     */
    public function getVPosRelTo()
    {
        return $this->vPosRelTo;
    }

    /**
     * Set vertical position relative to
     *
     * @param string $value
     * @return self
     */
    public function setVPosRelTo($value)
    {
        $enum = array(
            self::POS_RELTO_MARGIN,
            self::POS_RELTO_PAGE,
            self::POS_RELTO_TEXT,
            self::POS_RELTO_LINE,
            self::POS_RELTO_TMARGIN,
            self::POS_RELTO_BMARGIN,
            self::POS_RELTO_IMARGIN,
            self::POS_RELTO_OMARGIN,
        );
        $this->vPosRelTo = $this->setEnumVal($value, $enum, $this->vPosRelTo);

        return $this;
    }

    /**
     * Get wrap type
     *
     * @return string
     */
    public function getWrap()
    {
        return $this->wrap;
    }

    /**
     * Set wrap type
     *
     * @param string $value
     * @return self
     */
    public function setWrap($value)
    {
        $enum = array(
            self::WRAP_INLINE,
            self::WRAP_SQUARE,
            self::WRAP_TIGHT,
            self::WRAP_THROUGH,
            self::WRAP_TOPBOTTOM,
            self::WRAP_BEHIND,
            self::WRAP_INFRONT,
        );
        $this->wrap = $this->setEnumVal($value, $enum, $this->wrap);

        return $this;
    }

    /**
     * Get top distance from text wrap
     *
     * @return float
     */
    public function getWrapDistanceTop()
    {
        return $this->wrapDistanceTop;
    }

    /**
     * Set top distance from text wrap
     *
     * @param int $value
     * @return self
     */
    public function setWrapDistanceTop($value = null)
    {
        $this->wrapDistanceTop = $this->setFloatVal($value, null);

        return $this;
    }

    /**
     * Get bottom distance from text wrap
     *
     * @return float
     */
    public function getWrapDistanceBottom()
    {
        return $this->wrapDistanceBottom;
    }

    /**
     * Set bottom distance from text wrap
     *
     * @param float $value
     * @return self
     */
    public function setWrapDistanceBottom($value = null)
    {
        $this->wrapDistanceBottom = $this->setFloatVal($value, null);

        return $this;
    }

    /**
     * Get left distance from text wrap
     *
     * @return float
     */
    public function getWrapDistanceLeft()
    {
        return $this->wrapDistanceLeft;
    }

    /**
     * Set left distance from text wrap
     *
     * @param float $value
     * @return self
     */
    public function setWrapDistanceLeft($value = null)
    {
        $this->wrapDistanceLeft = $this->setFloatVal($value, null);

        return $this;
    }

    /**
     * Get right distance from text wrap
     *
     * @return float
     */
    public function getWrapDistanceRight()
    {
        return $this->wrapDistanceRight;
    }

    /**
     * Set right distance from text wrap
     *
     * @param float $value
     * @return self
     */
    public function setWrapDistanceRight($value = null)
    {
        $this->wrapDistanceRight = $this->setFloatVal($value, null);

        return $this;
    }

    /**
     * Get position
     *
     * @return int
     */
    public function getPosition()
    {
        return $this->position;
    }

    /**
     * Set position
     *
     * @param int $value
     * @return self
     */
    public function setPosition($value = null)
    {
        $this->position = $this->setIntVal($value, null);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Image and memory image style
 */
class Image extends Frame
{
    /**
     * Backward compatibility constants
     *
     * @const string
     */
    const WRAPPING_STYLE_INLINE = self::WRAP_INLINE;
    const WRAPPING_STYLE_SQUARE = self::WRAP_SQUARE;
    const WRAPPING_STYLE_TIGHT = self::WRAP_TIGHT;
    const WRAPPING_STYLE_BEHIND = self::WRAP_BEHIND;
    const WRAPPING_STYLE_INFRONT = self::WRAP_INFRONT;
    const POSITION_HORIZONTAL_LEFT = self::POS_LEFT;
    const POSITION_HORIZONTAL_CENTER = self::POS_CENTER;
    const POSITION_HORIZONTAL_RIGHT = self::POS_RIGHT;
    const POSITION_VERTICAL_TOP = self::POS_TOP;
    const POSITION_VERTICAL_CENTER = self::POS_CENTER;
    const POSITION_VERTICAL_BOTTOM = self::POS_BOTTOM;
    const POSITION_VERTICAL_INSIDE = self::POS_INSIDE;
    const POSITION_VERTICAL_OUTSIDE = self::POS_OUTSIDE;
    const POSITION_RELATIVE_TO_MARGIN = self::POS_RELTO_MARGIN;
    const POSITION_RELATIVE_TO_PAGE = self::POS_RELTO_PAGE;
    const POSITION_RELATIVE_TO_COLUMN = self::POS_RELTO_COLUMN;
    const POSITION_RELATIVE_TO_CHAR = self::POS_RELTO_CHAR;
    const POSITION_RELATIVE_TO_TEXT = self::POS_RELTO_TEXT;
    const POSITION_RELATIVE_TO_LINE = self::POS_RELTO_LINE;
    const POSITION_RELATIVE_TO_LMARGIN = self::POS_RELTO_LMARGIN;
    const POSITION_RELATIVE_TO_RMARGIN = self::POS_RELTO_RMARGIN;
    const POSITION_RELATIVE_TO_TMARGIN = self::POS_RELTO_TMARGIN;
    const POSITION_RELATIVE_TO_BMARGIN = self::POS_RELTO_BMARGIN;
    const POSITION_RELATIVE_TO_IMARGIN = self::POS_RELTO_IMARGIN;
    const POSITION_RELATIVE_TO_OMARGIN = self::POS_RELTO_OMARGIN;
    const POSITION_ABSOLUTE = self::POS_ABSOLUTE;
    const POSITION_RELATIVE = self::POS_RELATIVE;

    /**
     * Create new instance
     */
    public function __construct()
    {
        parent::__construct();
        $this->setUnit(self::UNIT_PT);

        // Backward compatibility setting
        // @todo Remove on 1.0.0
        $this->setWrap(self::WRAPPING_STYLE_INLINE);
        $this->setHPos(self::POSITION_HORIZONTAL_LEFT);
        $this->setHPosRelTo(self::POSITION_RELATIVE_TO_CHAR);
        $this->setVPos(self::POSITION_VERTICAL_TOP);
        $this->setVPosRelTo(self::POSITION_RELATIVE_TO_LINE);
    }

    /**
     * Get margin top
     *
     * @return int|float
     */
    public function getMarginTop()
    {
        return $this->getTop();
    }

    /**
     * Set margin top
     *
     * @ignoreScrutinizerPatch
     * @param int|float $value
     * @return self
     */
    public function setMarginTop($value = 0)
    {
        $this->setTop($value);

        return $this;
    }

    /**
     * Get margin left
     *
     * @return int|float
     */
    public function getMarginLeft()
    {
        return $this->getLeft();
    }

    /**
     * Set margin left
     *
     * @ignoreScrutinizerPatch
     * @param int|float $value
     * @return self
     */
    public function setMarginLeft($value = 0)
    {
        $this->setLeft($value);

        return $this;
    }

    /**
     * Get wrapping style
     *
     * @return string
     */
    public function getWrappingStyle()
    {
        return $this->getWrap();
    }

    /**
     * Set wrapping style
     *
     * @param string $wrappingStyle
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setWrappingStyle($wrappingStyle)
    {
        $this->setWrap($wrappingStyle);

        return $this;
    }

    /**
     * Get positioning type
     *
     * @return string
     */
    public function getPositioning()
    {
        return $this->getPos();
    }

    /**
     * Set positioning type
     *
     * @param string $positioning
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setPositioning($positioning)
    {
        $this->setPos($positioning);

        return $this;
    }

    /**
     * Get horizontal alignment
     *
     * @return string
     */
    public function getPosHorizontal()
    {
        return $this->getHPos();
    }

    /**
     * Set horizontal alignment
     *
     * @param string $alignment
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setPosHorizontal($alignment)
    {
        $this->setHPos($alignment);

        return $this;
    }

    /**
     * Get vertical alignment
     *
     * @return string
     */
    public function getPosVertical()
    {
        return $this->getVPos();
    }

    /**
     * Set vertical alignment
     *
     * @param string $alignment
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setPosVertical($alignment)
    {
        $this->setVPos($alignment);

        return $this;
    }

    /**
     * Get horizontal relation
     *
     * @return string
     */
    public function getPosHorizontalRel()
    {
        return $this->getHPosRelTo();
    }

    /**
     * Set horizontal relation
     *
     * @param string $relto
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setPosHorizontalRel($relto)
    {
        $this->setHPosRelTo($relto);

        return $this;
    }

    /**
     * Get vertical relation
     *
     * @return string
     */
    public function getPosVerticalRel()
    {
        return $this->getVPosRelTo();
    }

    /**
     * Set vertical relation
     *
     * @param string $relto
     *
     * @throws \InvalidArgumentException
     *
     * @return self
     */
    public function setPosVerticalRel($relto)
    {
        $this->setVPosRelTo($relto);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Indentation.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Paragraph indentation style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_Ind.html
 * @since 0.10.0
 */
class Indentation extends AbstractStyle
{
    /**
     * Left indentation (twip)
     *
     * @var int|float
     */
    private $left = 0;

    /**
     * Right indentation (twip)
     *
     * @var int|float
     */
    private $right = 0;

    /**
     * Additional first line indentation (twip)
     *
     * @var int|float
     */
    private $firstLine;

    /**
     * Indentation removed from first line (twip)
     *
     * @var int|float
     */
    private $hanging;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get left
     *
     * @return int|float
     */
    public function getLeft()
    {
        return $this->left;
    }

    /**
     * Set left
     *
     * @param int|float $value
     * @return self
     */
    public function setLeft($value = null)
    {
        $this->left = $this->setNumericVal($value, $this->left);

        return $this;
    }

    /**
     * Get right
     *
     * @return int|float
     */
    public function getRight()
    {
        return $this->right;
    }

    /**
     * Set right
     *
     * @param int|float $value
     * @return self
     */
    public function setRight($value = null)
    {
        $this->right = $this->setNumericVal($value, $this->right);

        return $this;
    }

    /**
     * Get first line
     *
     * @return int|float
     */
    public function getFirstLine()
    {
        return $this->firstLine;
    }

    /**
     * Set first line
     *
     * @param int|float $value
     * @return self
     */
    public function setFirstLine($value = null)
    {
        $this->firstLine = $this->setNumericVal($value, $this->firstLine);

        return $this;
    }

    /**
     * Get hanging
     *
     * @return int|float
     */
    public function getHanging()
    {
        return $this->hanging;
    }

    /**
     * Set hanging
     *
     * @param int|float $value
     * @return self
     */
    public function setHanging($value = null)
    {
        $this->hanging = $this->setNumericVal($value, $this->hanging);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Language.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Language
 * A couple of predefined values are defined here, see the websites below for more values
 *
 * @see http://www.datypic.com/sc/ooxml/t-w_CT_Language.html
 * @see https://technet.microsoft.com/en-us/library/cc287874(v=office.12).aspx
 */
final class Language extends AbstractStyle
{
    const EN_US = 'en-US';
    const EN_US_ID = 1033;

    const EN_GB = 'en-GB';
    const EN_GB_ID = 2057;

    const FR_FR = 'fr-FR';
    const FR_FR_ID = 1036;

    const FR_BE = 'fr-BE';
    const FR_BE_ID = 2060;

    const ES_ES = 'es-ES';
    const ES_ES_ID = 3082;

    const DE_DE = 'de-DE';
    const DE_DE_ID = 1031;

    const HE_IL = 'he-IL';
    const HE_IL_ID = 1037;

    const IT_IT = 'it-IT';
    const IT_IT_ID = 1040;

    const JA_JP = 'ja-JP';
    const JA_JP_ID = 1041;

    const KO_KR = 'ko-KR';
    const KO_KR_ID = 1042;

    const ZH_CN = 'zh-CN';
    const ZH_CN_ID = 2052;

    const HI_IN = 'hi-IN';
    const HI_IN_ID = 1081;

    const PT_BR = 'pt-BR';
    const PT_BR_ID = 1046;

    const NL_NL = 'nl-NL';
    const NL_NL_ID = 1043;

    const UK_UA = 'uk-UA';
    const UK_UA_ID = 1058;

    const RU_RU = 'ru-RU';
    const RU_RU_ID = 1049;

    /**
     * Language ID, used for RTF document generation
     *
     * @var int
     * @see https://technet.microsoft.com/en-us/library/cc179219.aspx
     */
    private $langId;

    /**
     * Latin Language
     *
     * @var string
     */
    private $latin;

    /**
     * East Asian Language
     *
     * @var string
     */
    private $eastAsia;

    /**
     * Complex Script Language
     *
     * @var string
     */
    private $bidirectional;

    /**
     * Constructor
     *
     * @param string|null $latin
     * @param string|null $eastAsia
     * @param string|null $bidirectional
     */
    public function __construct($latin = null, $eastAsia = null, $bidirectional = null)
    {
        if (!empty($latin)) {
            $this->setLatin($latin);
        }
        if (!empty($eastAsia)) {
            $this->setEastAsia($eastAsia);
        }
        if (!empty($bidirectional)) {
            $this->setBidirectional($bidirectional);
        }
    }

    /**
     * Set the Latin Language
     *
     * @param string $latin
     *            The value for the latin language
     * @return self
     */
    public function setLatin($latin)
    {
        $this->latin = $this->validateLocale($latin);

        return $this;
    }

    /**
     * Get the Latin Language
     *
     * @return string|null
     */
    public function getLatin()
    {
        return $this->latin;
    }

    /**
     * Set the Language ID
     *
     * @param int $langId
     *            The value for the language ID
     * @return self
     * @see https://technet.microsoft.com/en-us/library/cc287874(v=office.12).aspx
     */
    public function setLangId($langId)
    {
        $this->langId = $langId;

        return $this;
    }

    /**
     * Get the Language ID
     *
     * @return int
     */
    public function getLangId()
    {
        return $this->langId;
    }

    /**
     * Set the East Asian Language
     *
     * @param string $eastAsia
     *            The value for the east asian language
     * @return self
     */
    public function setEastAsia($eastAsia)
    {
        $this->eastAsia = $this->validateLocale($eastAsia);

        return $this;
    }

    /**
     * Get the East Asian Language
     *
     * @return string|null
     */
    public function getEastAsia()
    {
        return $this->eastAsia;
    }

    /**
     * Set the Complex Script Language
     *
     * @param string $bidirectional
     *            The value for the complex script language
     * @return self
     */
    public function setBidirectional($bidirectional)
    {
        $this->bidirectional = $this->validateLocale($bidirectional);

        return $this;
    }

    /**
     * Get the Complex Script Language
     *
     * @return string|null
     */
    public function getBidirectional()
    {
        return $this->bidirectional;
    }

    /**
     * Validates that the language passed is in the format xx-xx
     *
     * @param string $locale
     * @return string
     */
    private function validateLocale($locale)
    {
        if ($locale !== null) {
            $locale = str_replace('_', '-', $locale);
        }

        if (strlen($locale) === 2) {
            return strtolower($locale) . '-' . strtoupper($locale);
        }

        if ($locale !== null && $locale !== 'zxx' && strstr($locale, '-') === false) {
            throw new \InvalidArgumentException($locale . ' is not a valid language code');
        }

        return $locale;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Line.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Line style
 */
class Line extends Image
{
    /**
     * Connector types
     *
     * @const string
     */
    const CONNECTOR_TYPE_STRAIGHT = 'straight';

    /**
     * Arrow styles
     *
     * @const string
     */
    const ARROW_STYLE_BLOCK = 'block';
    const ARROW_STYLE_OPEN = 'open';
    const ARROW_STYLE_CLASSIC = 'classic';
    const ARROW_STYLE_DIAMOND = 'diamond';
    const ARROW_STYLE_OVAL = 'oval';

    /**
     * Dash styles
     *
     * @const string
     */
    const DASH_STYLE_DASH = 'dash';
    const DASH_STYLE_ROUND_DOT = 'rounddot';
    const DASH_STYLE_SQUARE_DOT = 'squaredot';
    const DASH_STYLE_DASH_DOT = 'dashdot';
    const DASH_STYLE_LONG_DASH = 'longdash';
    const DASH_STYLE_LONG_DASH_DOT = 'longdashdot';
    const DASH_STYLE_LONG_DASH_DOT_DOT = 'longdashdotdot';

    /**
     * flip Line
     *
     * @var bool
     */
    private $flip = false;

    /**
     * connectorType
     *
     * @var string
     */
    private $connectorType = self::CONNECTOR_TYPE_STRAIGHT;

    /**
     * Line Weight
     *
     * @var int
     */
    private $weight;

    /**
     * Line color
     *
     * @var string
     */
    private $color;

    /**
     * Dash style
     *
     * @var string
     */
    private $dash;

    /**
     * Begin arrow
     *
     * @var string
     */
    private $beginArrow;

    /**
     * End arrow
     *
     * @var string
     */
    private $endArrow;

    /**
     * Get flip
     *
     * @return bool
     */
    public function isFlip()
    {
        return $this->flip;
    }

    /**
     * Set flip
     *
     * @param bool $value
     * @return self
     */
    public function setFlip($value = false)
    {
        $this->flip = $this->setBoolVal($value, $this->flip);

        return $this;
    }

    /**
     * Get connectorType
     *
     * @return string
     */
    public function getConnectorType()
    {
        return $this->connectorType;
    }

    /**
     * Set connectorType
     *
     * @param string $value
     * @return self
     */
    public function setConnectorType($value = null)
    {
        $enum = array(
            self::CONNECTOR_TYPE_STRAIGHT,
        );
        $this->connectorType = $this->setEnumVal($value, $enum, $this->connectorType);

        return $this;
    }

    /**
     * Get weight
     *
     * @return int
     */
    public function getWeight()
    {
        return $this->weight;
    }

    /**
     * Set weight
     *
     * @param int $value Weight in points
     * @return self
     */
    public function setWeight($value = null)
    {
        $this->weight = $this->setNumericVal($value, $this->weight);

        return $this;
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }

    /**
     * Get beginArrow
     *
     * @return string
     */
    public function getBeginArrow()
    {
        return $this->beginArrow;
    }

    /**
     * Set beginArrow
     *
     * @param string $value
     * @return self
     */
    public function setBeginArrow($value = null)
    {
        $enum = array(
            self::ARROW_STYLE_BLOCK, self::ARROW_STYLE_CLASSIC, self::ARROW_STYLE_DIAMOND,
            self::ARROW_STYLE_OPEN, self::ARROW_STYLE_OVAL,
        );
        $this->beginArrow = $this->setEnumVal($value, $enum, $this->beginArrow);

        return $this;
    }

    /**
     * Get endArrow
     *
     * @return string
     */
    public function getEndArrow()
    {
        return $this->endArrow;
    }

    /**
     * Set endArrow
     *
     * @param string $value
     * @return self
     */
    public function setEndArrow($value = null)
    {
        $enum = array(
            self::ARROW_STYLE_BLOCK, self::ARROW_STYLE_CLASSIC, self::ARROW_STYLE_DIAMOND,
            self::ARROW_STYLE_OPEN, self::ARROW_STYLE_OVAL,
        );
        $this->endArrow = $this->setEnumVal($value, $enum, $this->endArrow);

        return $this;
    }

    /**
     * Get Dash
     *
     * @return string
     */
    public function getDash()
    {
        return $this->dash;
    }

    /**
     * Set Dash
     *
     * @param string $value
     * @return self
     */
    public function setDash($value = null)
    {
        $enum = array(
            self::DASH_STYLE_DASH, self::DASH_STYLE_DASH_DOT, self::DASH_STYLE_LONG_DASH,
            self::DASH_STYLE_LONG_DASH_DOT, self::DASH_STYLE_LONG_DASH_DOT_DOT, self::DASH_STYLE_ROUND_DOT,
            self::DASH_STYLE_SQUARE_DOT,
        );
        $this->dash = $this->setEnumVal($value, $enum, $this->dash);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/LineNumbering.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Line numbering style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_LineNumber.html
 * @since 0.10.0
 */
class LineNumbering extends AbstractStyle
{
    /** @const string Line numbering restart setting http://www.schemacentral.com/sc/ooxml/a-w_restart-1.html */
    const LINE_NUMBERING_CONTINUOUS = 'continuous';
    const LINE_NUMBERING_NEW_PAGE = 'newPage';
    const LINE_NUMBERING_NEW_SECTION = 'newSection';

    /**
     * Line numbering starting value
     *
     * @var int
     */
    private $start = 1;

    /**
     * Line number increments
     *
     * @var int
     */
    private $increment = 1;

    /**
     * Distance between text and line numbering in twip
     *
     * @var int|float
     */
    private $distance;

    /**
     * Line numbering restart setting continuous|newPage|newSection
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/a-w_restart-1.html
     */
    private $restart;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get start
     *
     * @return int
     */
    public function getStart()
    {
        return $this->start;
    }

    /**
     * Set start
     *
     * @param int $value
     * @return self
     */
    public function setStart($value = null)
    {
        $this->start = $this->setIntVal($value, $this->start);

        return $this;
    }

    /**
     * Get increment
     *
     * @return int
     */
    public function getIncrement()
    {
        return $this->increment;
    }

    /**
     * Set increment
     *
     * @param int $value
     * @return self
     */
    public function setIncrement($value = null)
    {
        $this->increment = $this->setIntVal($value, $this->increment);

        return $this;
    }

    /**
     * Get distance
     *
     * @return int|float
     */
    public function getDistance()
    {
        return $this->distance;
    }

    /**
     * Set distance
     *
     * @param int|float $value
     * @return self
     */
    public function setDistance($value = null)
    {
        $this->distance = $this->setNumericVal($value, $this->distance);

        return $this;
    }

    /**
     * Get restart
     *
     * @return string
     */
    public function getRestart()
    {
        return $this->restart;
    }

    /**
     * Set distance
     *
     * @param string $value
     * @return self
     */
    public function setRestart($value = null)
    {
        $enum = array(self::LINE_NUMBERING_CONTINUOUS, self::LINE_NUMBERING_NEW_PAGE, self::LINE_NUMBERING_NEW_SECTION);
        $this->restart = $this->setEnumVal($value, $enum, $this->restart);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Numbering.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Numbering style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/e-w_numbering.html
 * @see  http://www.schemacentral.com/sc/ooxml/e-w_abstractNum-1.html
 * @see  http://www.schemacentral.com/sc/ooxml/e-w_num-1.html
 * @since 0.10.0
 */
class Numbering extends AbstractStyle
{
    /**
     * Numbering definition instance ID
     *
     * @var int
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_num-1.html
     */
    private $numId;

    /**
     * Multilevel type singleLevel|multilevel|hybridMultilevel
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/a-w_val-67.html
     */
    private $type;

    /**
     * Numbering levels
     *
     * @var NumberingLevel[]
     */
    private $levels = array();

    /**
     * Get Id
     *
     * @return int
     */
    public function getNumId()
    {
        return $this->numId;
    }

    /**
     * Set Id
     *
     * @param int $value
     * @return self
     */
    public function setNumId($value)
    {
        $this->numId = $this->setIntVal($value, $this->numId);

        return $this;
    }

    /**
     * Get multilevel type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set multilevel type
     *
     * @param string $value
     * @return self
     */
    public function setType($value)
    {
        $enum = array('singleLevel', 'multilevel', 'hybridMultilevel');
        $this->type = $this->setEnumVal($value, $enum, $this->type);

        return $this;
    }

    /**
     * Get levels
     *
     * @return NumberingLevel[]
     */
    public function getLevels()
    {
        return $this->levels;
    }

    /**
     * Set multilevel type
     *
     * @param array $values
     * @return self
     */
    public function setLevels($values)
    {
        if (is_array($values)) {
            foreach ($values as $key => $value) {
                $numberingLevel = new NumberingLevel();
                if (is_array($value)) {
                    $numberingLevel->setStyleByArray($value);
                    $numberingLevel->setLevel($key);
                }
                $this->levels[$key] = $numberingLevel;
            }
        }

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/NumberingLevel.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\SimpleType\Jc;
use PhpOffice\PhpWord\SimpleType\NumberFormat;

/**
 * Numbering level definition
 *
 * @see  http://www.schemacentral.com/sc/ooxml/e-w_lvl-1.html
 * @since 0.10.0
 */
class NumberingLevel extends AbstractStyle
{
    /**
     * Level number, 0 to 8 (total 9 levels)
     *
     * @var int
     */
    private $level = 0;

    /**
     * Starting value w:start
     *
     * @var int
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_start-1.html
     */
    private $start = 1;

    /**
     * Numbering format w:numFmt, one of PhpOffice\PhpWord\SimpleType\NumberFormat
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/t-w_ST_NumberFormat.html
     */
    private $format;

    /**
     * Restart numbering level symbol w:lvlRestart
     *
     * @var int
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_lvlRestart-1.html
     */
    private $restart;

    /**
     * Related paragraph style
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_pStyle-2.html
     */
    private $pStyle;

    /**
     * Content between numbering symbol and paragraph text w:suff
     *
     * @var string tab|space|nothing
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_suff-1.html
     */
    private $suffix = 'tab';

    /**
     * Numbering level text e.g. %1 for nonbullet or bullet character
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_lvlText-1.html
     */
    private $text;

    /**
     * Justification, w:lvlJc
     *
     * @var string, one of PhpOffice\PhpWord\SimpleType\Jc
     */
    private $alignment = '';

    /**
     * Left
     *
     * @var int
     */
    private $left;

    /**
     * Hanging
     *
     * @var int
     */
    private $hanging;

    /**
     * Tab position
     *
     * @var int
     */
    private $tabPos;

    /**
     * Font family
     *
     * @var string
     */
    private $font;

    /**
     * Hint default|eastAsia|cs
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/a-w_hint-1.html
     */
    private $hint;

    /**
     * Get level
     *
     * @return int
     */
    public function getLevel()
    {
        return $this->level;
    }

    /**
     * Set level
     *
     * @param int $value
     * @return self
     */
    public function setLevel($value)
    {
        $this->level = $this->setIntVal($value, $this->level);

        return $this;
    }

    /**
     * Get start
     *
     * @return int
     */
    public function getStart()
    {
        return $this->start;
    }

    /**
     * Set start
     *
     * @param int $value
     * @return self
     */
    public function setStart($value)
    {
        $this->start = $this->setIntVal($value, $this->start);

        return $this;
    }

    /**
     * Get format
     *
     * @return string
     */
    public function getFormat()
    {
        return $this->format;
    }

    /**
     * Set format
     *
     * @param string $value
     * @return self
     */
    public function setFormat($value)
    {
        $this->format = $this->setEnumVal($value, NumberFormat::values(), $this->format);

        return $this;
    }

    /**
     * Get restart
     *
     * @return int
     */
    public function getRestart()
    {
        return $this->restart;
    }

    /**
     * Set restart
     *
     * @param int $value
     * @return self
     */
    public function setRestart($value)
    {
        $this->restart = $this->setIntVal($value, $this->restart);

        return $this;
    }

    /**
     * Get related paragraph style
     *
     * @return string
     */
    public function getPStyle()
    {
        return $this->pStyle;
    }

    /**
     * Set  related paragraph style
     *
     * @param string $value
     * @return self
     */
    public function setPStyle($value)
    {
        $this->pStyle = $value;

        return $this;
    }

    /**
     * Get suffix
     *
     * @return string
     */
    public function getSuffix()
    {
        return $this->suffix;
    }

    /**
     * Set suffix
     *
     * @param string $value
     * @return self
     */
    public function setSuffix($value)
    {
        $enum = array('tab', 'space', 'nothing');
        $this->suffix = $this->setEnumVal($value, $enum, $this->suffix);

        return $this;
    }

    /**
     * Get text
     *
     * @return string
     */
    public function getText()
    {
        return $this->text;
    }

    /**
     * Set text
     *
     * @param string $value
     * @return self
     */
    public function setText($value)
    {
        $this->text = $value;

        return $this;
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    public function getAlignment()
    {
        return $this->alignment;
    }

    /**
     * @since 0.13.0
     *
     * @param string $value
     *
     * @return self
     */
    public function setAlignment($value)
    {
        if (Jc::isValid($value)) {
            $this->alignment = $value;
        }

        return $this;
    }

    /**
     * @deprecated 0.13.0 Use the `getAlignment` method instead.
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getAlign()
    {
        return $this->getAlignment();
    }

    /**
     * @deprecated 0.13.0 Use the `setAlignment` method instead.
     *
     * @param string $value
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setAlign($value)
    {
        return $this->setAlignment($value);
    }

    /**
     * Get left
     *
     * @return int
     */
    public function getLeft()
    {
        return $this->left;
    }

    /**
     * Set left
     *
     * @param int $value
     * @return self
     */
    public function setLeft($value)
    {
        $this->left = $this->setIntVal($value, $this->left);

        return $this;
    }

    /**
     * Get hanging
     *
     * @return int
     */
    public function getHanging()
    {
        return $this->hanging;
    }

    /**
     * Set hanging
     *
     * @param int $value
     * @return self
     */
    public function setHanging($value)
    {
        $this->hanging = $this->setIntVal($value, $this->hanging);

        return $this;
    }

    /**
     * Get tab
     *
     * @return int
     */
    public function getTabPos()
    {
        return $this->tabPos;
    }

    /**
     * Set tab
     *
     * @param int $value
     * @return self
     */
    public function setTabPos($value)
    {
        $this->tabPos = $this->setIntVal($value, $this->tabPos);

        return $this;
    }

    /**
     * Get font
     *
     * @return string
     */
    public function getFont()
    {
        return $this->font;
    }

    /**
     * Set font
     *
     * @param string $value
     * @return self
     */
    public function setFont($value)
    {
        $this->font = $value;

        return $this;
    }

    /**
     * Get hint
     *
     * @return string
     */
    public function getHint()
    {
        return $this->hint;
    }

    /**
     * Set hint
     *
     * @param string $value
     * @return self
     */
    public function setHint($value = null)
    {
        $enum = array('default', 'eastAsia', 'cs');
        $this->hint = $this->setEnumVal($value, $enum, $this->hint);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Outline.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Outline defines the line/border of the object
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-v_CT_Stroke.html
 * @see  http://www.w3.org/TR/1998/NOTE-VML-19980513#_Toc416858395
 * @since 0.12.0
 */
class Outline extends AbstractStyle
{
    /**
     * Line style constants
     *
     * @see  http://www.schemacentral.com/sc/ooxml/t-v_ST_StrokeLineStyle.html
     * @const string
     */
    const LINE_SINGLE = 'single';
    const LINE_THIN_THIN = 'thinThin';
    const LINE_THIN_THICK = 'thinThick';
    const LINE_THICK_THIN = 'thickThin';
    const LINE_THICK_BETWEEN_THIN = 'thickBetweenThin';

    /**
     * Line style constants
     *
     * @see  http://www.schemacentral.com/sc/ooxml/t-v_ST_StrokeEndCap.html
     * @const string
     */
    const ENDCAP_FLAT = 'flat';
    const ENDCAP_SQUARE = 'square';
    const ENDCAP_ROUND = 'round';

    /**
     * Arrowhead type constants
     *
     * @see  http://www.schemacentral.com/sc/ooxml/t-v_ST_StrokeArrowType.html
     * @const string
     */
    const ARROW_NONE = 'none';
    const ARROW_BLOCK = 'block';
    const ARROW_CLASSIC = 'classic';
    const ARROW_OVAL = 'oval';
    const ARROW_DIAMOND = 'diamond';
    const ARROW_OPEN = 'open';

    /**
     * Unit; No set method for now
     *
     * @var string
     */
    private $unit = 'pt';

    /**
     * Outline weight
     *
     * @var int|float
     */
    private $weight;

    /**
     * Outline color
     *
     * @var string
     */
    private $color;

    /**
     * Dash type
     *
     * @var string
     */
    private $dash;

    /**
     * Line style
     *
     * @var string
     */
    private $line;

    /**
     * End cap
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/t-v_ST_StrokeEndCap.html
     */
    private $endCap;

    /**
     * Start arrow type
     *
     * @var string
     */
    private $startArrow;

    /**
     * End arrow type
     *
     * @var string
     */
    private $endArrow;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get unit
     *
     * @return string
     */
    public function getUnit()
    {
        return $this->unit;
    }

    /**
     * Get weight
     *
     * @return int|float
     */
    public function getWeight()
    {
        return $this->weight;
    }

    /**
     * Set weight
     *
     * @param int|float $value
     * @return self
     */
    public function setWeight($value = null)
    {
        $this->weight = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }

    /**
     * Get dash type
     *
     * @return string
     */
    public function getDash()
    {
        return $this->dash;
    }

    /**
     * Set dash type
     *
     * @param string $value
     * @return self
     */
    public function setDash($value = null)
    {
        $this->dash = $value;

        return $this;
    }

    /**
     * Get line style
     *
     * @return string
     */
    public function getLine()
    {
        return $this->line;
    }

    /**
     * Set line style
     *
     * @param string $value
     * @return self
     */
    public function setLine($value = null)
    {
        $enum = array(self::LINE_SINGLE, self::LINE_THIN_THIN, self::LINE_THIN_THICK,
            self::LINE_THICK_THIN, self::LINE_THICK_BETWEEN_THIN, );
        $this->line = $this->setEnumVal($value, $enum, null);

        return $this;
    }

    /**
     * Get endCap style
     *
     * @return string
     */
    public function getEndCap()
    {
        return $this->endCap;
    }

    /**
     * Set endCap style
     *
     * @param string $value
     * @return self
     */
    public function setEndCap($value = null)
    {
        $enum = array(self::ENDCAP_FLAT, self::ENDCAP_SQUARE, self::ENDCAP_ROUND);
        $this->endCap = $this->setEnumVal($value, $enum, null);

        return $this;
    }

    /**
     * Get startArrow
     *
     * @return string
     */
    public function getStartArrow()
    {
        return $this->startArrow;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setStartArrow($value = null)
    {
        $enum = array(self::ARROW_NONE, self::ARROW_BLOCK, self::ARROW_CLASSIC,
            self::ARROW_OVAL, self::ARROW_DIAMOND, self::ARROW_OPEN, );
        $this->startArrow = $this->setEnumVal($value, $enum, null);

        return $this;
    }

    /**
     * Get endArrow
     *
     * @return string
     */
    public function getEndArrow()
    {
        return $this->endArrow;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setEndArrow($value = null)
    {
        $enum = array(self::ARROW_NONE, self::ARROW_BLOCK, self::ARROW_CLASSIC,
            self::ARROW_OVAL, self::ARROW_DIAMOND, self::ARROW_OPEN, );
        $this->endArrow = $this->setEnumVal($value, $enum, null);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Paper.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\Shared\Converter;

/**
 * Paper size from ISO/IEC 29500-1:2012 pg. 1656-1657
 *
 * 1 = Letter paper (8.5 in. by 11 in.)
 * 2 = Letter small paper (8.5 in. by 11 in.)
 * 3 = Tabloid paper (11 in. by 17 in.)
 * 4 = Ledger paper (17 in. by 11 in.)
 * 5 = Legal paper (8.5 in. by 14 in.)
 * 6 = Statement paper (5.5 in. by 8.5 in.)
 * 7 = Executive paper (7.25 in. by 10.5 in.)
 * 8 = A3 paper (297 mm by 420 mm)
 * 9 = A4 paper (210 mm by 297 mm)
 * 10 = A4 small paper (210 mm by 297 mm)
 * 11 = A5 paper (148 mm by 210 mm)
 * 12 = B4 paper (250 mm by 353 mm)
 * 13 = B5 paper (176 mm by 250 mm)
 * 14 = Folio paper (8.5 in. by 13 in.)
 * 15 = Quarto paper (215 mm by 275 mm)
 * 16 = Standard paper (10 in. by 14 in.)
 * 17 = Standard paper (11 in. by 17 in.)
 * 18 = Note paper (8.5 in. by 11 in.)
 * 19 = #9 envelope (3.875 in. by 8.875 in.)
 * 20 = #10 envelope (4.125 in. by 9.5 in.)
 * 21 = #11 envelope (4.5 in. by 10.375 in.)
 * 22 = #12 envelope (4.75 in. by 11 in.)
 * 23 = #14 envelope (5 in. by 11.5 in.)
 * 24 = C paper (17 in. by 22 in.)
 * 25 = D paper (22 in. by 34 in.)
 * 26 = E paper (34 in. by 44 in.)
 * 27 = DL envelope (110 mm by 220 mm)
 * 28 = C5 envelope (162 mm by 229 mm)
 * 29 = C3 envelope (324 mm by 458 mm)
 * 30 = C4 envelope (229 mm by 324 mm)
 * 31 = C6 envelope (114 mm by 162 mm)
 * 32 = C65 envelope (114 mm by 229 mm)
 * 33 = B4 envelope (250 mm by 353 mm)
 * 34 = B5 envelope (176 mm by 250 mm)
 * 35 = B6 envelope (176 mm by 125 mm)
 * 36 = Italy envelope (110 mm by 230 mm)
 * 37 = Monarch envelope (3.875 in. by 7.5 in.).
 * 38 = 6 3/4 envelope (3.625 in. by 6.5 in.)
 * 39 = US standard fanfold (14.875 in. by 11 in.)
 * 40 = German standard fanfold (8.5 in. by 12 in.)
 * 41 = German legal fanfold (8.5 in. by 13 in.)
 * 42 = ISO B4 (250 mm by 353 mm)
 * 43 = Japanese double postcard (200 mm by 148 mm)
 * 44 = Standard paper (9 in. by 11 in.)
 * 45 = Standard paper (10 in. by 11 in.)
 * 46 = Standard paper (15 in. by 11 in.)
 * 47 = Invite envelope (220 mm by 220 mm)
 * 50 = Letter extra paper (9.275 in. by 12 in.)
 * 51 = Legal extra paper (9.275 in. by 15 in.)
 * 52 = Tabloid extra paper (11.69 in. by 18 in.)
 * 53 = A4 extra paper (236 mm by 322 mm)
 * 54 = Letter transverse paper (8.275 in. by 11 in.)
 * 55 = A4 transverse paper (210 mm by 297 mm)
 * 56 = Letter extra transverse paper (9.275 in. by 12 in.)
 * 57 = SuperA/SuperA/A4 paper (227 mm by 356 mm)
 * 58 = SuperB/SuperB/A3 paper (305 mm by 487 mm)
 * 59 = Letter plus paper (8.5 in. by 12.69 in.)
 * 60 = A4 plus paper (210 mm by 330 mm)
 * 61 = A5 transverse paper (148 mm by 210 mm)
 * 62 = JIS B5 transverse paper (182 mm by 257 mm)
 * 63 = A3 extra paper (322 mm by 445 mm)
 * 64 = A5 extra paper (174 mm by 235 mm)
 * 65 = ISO B5 extra paper (201 mm by 276 mm)
 * 66 = A2 paper (420 mm by 594 mm)
 * 67 = A3 transverse paper (297 mm by 420 mm)
 * 68 = A3 extra transverse paper (322 mm by 445 mm)
 *
 * @since 0.12.0
 */
class Paper extends AbstractStyle
{
    /**
     * Paper sizes
     *
     * @var array
     */
    private $sizes = array(
        'A3'        => array(297, 420, 'mm'),
        'A4'        => array(210, 297, 'mm'),
        'A5'        => array(148, 210, 'mm'),
        'B5'        => array(176, 250, 'mm'),
        'Folio'     => array(8.5, 13, 'in'),
        'Legal'     => array(8.5, 14, 'in'),
        'Letter'    => array(8.5, 11, 'in'),
    );

    /**
     * Paper size
     *
     * @var string
     */
    private $size = 'A4';

    /**
     * Width
     *
     * @var float (twip)
     */
    private $width;

    /**
     * Height
     *
     * @var float (twip)
     */
    private $height;

    /**
     * Create a new instance
     *
     * @param string $size
     */
    public function __construct($size = 'A4')
    {
        $this->setSize($size);
    }

    /**
     * Get size
     *
     * @return string
     */
    public function getSize()
    {
        return $this->size;
    }

    /**
     * Set size
     *
     * @param string $size
     * @return self
     */
    public function setSize($size)
    {
        $this->size = $this->setEnumVal($size, array_keys($this->sizes), $this->size);

        list($width, $height, $unit) = $this->sizes[$this->size];

        if ($unit == 'mm') {
            $this->width = Converter::cmToTwip($width / 10);
            $this->height = Converter::cmToTwip($height / 10);
        } else {
            $this->width = Converter::inchToTwip($width);
            $this->height = Converter::inchToTwip($height);
        }

        return $this;
    }

    /**
     * Get width
     *
     * @return float
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Get height
     *
     * @return float
     */
    public function getHeight()
    {
        return $this->height;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Paragraph.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\Exception\InvalidStyleException;
use PhpOffice\PhpWord\Shared\Text;
use PhpOffice\PhpWord\SimpleType\Jc;
use PhpOffice\PhpWord\SimpleType\TextAlignment;

/**
 * Paragraph style
 *
 * OOXML:
 * - General: alignment, outline level
 * - Indentation: left, right, firstline, hanging
 * - Spacing: before, after, line spacing
 * - Pagination: widow control, keep next, keep line, page break before
 * - Formatting exception: suppress line numbers, don't hyphenate
 * - Textbox options
 * - Tabs
 * - Shading
 * - Borders
 *
 * OpenOffice:
 * - Indents & spacing
 * - Alignment
 * - Text flow
 * - Outline & numbering
 * - Tabs
 * - Dropcaps
 * - Tabs
 * - Borders
 * - Background
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_PPr.html
 */
class Paragraph extends Border
{
    /**
     * @const int One line height equals 240 twip
     */
    const LINE_HEIGHT = 240;

    /**
     * Aliases
     *
     * @var array
     */
    protected $aliases = array('line-height' => 'lineHeight', 'line-spacing' => 'spacing');

    /**
     * Parent style
     *
     * @var string
     */
    private $basedOn = 'Normal';

    /**
     * Style for next paragraph
     *
     * @var string
     */
    private $next;

    /**
     * @var string
     */
    private $alignment = '';

    /**
     * Indentation
     *
     * @var \PhpOffice\PhpWord\Style\Indentation|null
     */
    private $indentation;

    /**
     * Spacing
     *
     * @var \PhpOffice\PhpWord\Style\Spacing
     */
    private $spacing;

    /**
     * Text line height
     *
     * @var int
     */
    private $lineHeight;

    /**
     * Allow first/last line to display on a separate page
     *
     * @var bool
     */
    private $widowControl = true;

    /**
     * Keep paragraph with next paragraph
     *
     * @var bool
     */
    private $keepNext = false;

    /**
     * Keep all lines on one page
     *
     * @var bool
     */
    private $keepLines = false;

    /**
     * Start paragraph on next page
     *
     * @var bool
     */
    private $pageBreakBefore = false;

    /**
     * Numbering style name
     *
     * @var string
     */
    private $numStyle;

    /**
     * Numbering level
     *
     * @var int
     */
    private $numLevel = 0;

    /**
     * Set of Custom Tab Stops
     *
     * @var \PhpOffice\PhpWord\Style\Tab[]
     */
    private $tabs = array();

    /**
     * Shading
     *
     * @var \PhpOffice\PhpWord\Style\Shading
     */
    private $shading;

    /**
     * Ignore Spacing Above and Below When Using Identical Styles
     *
     * @var bool
     */
    private $contextualSpacing = false;

    /**
     * Right to Left Paragraph Layout
     *
     * @var bool
     */
    private $bidi = false;

    /**
     * Vertical Character Alignment on Line
     *
     * @var string
     */
    private $textAlignment;

    /**
     * Suppress hyphenation for paragraph
     *
     * @var bool
     */
    private $suppressAutoHyphens = false;

    /**
     * Set Style value
     *
     * @param string $key
     * @param mixed $value
     * @return self
     */
    public function setStyleValue($key, $value)
    {
        $key = Text::removeUnderscorePrefix($key);
        if ('indent' == $key || 'hanging' == $key) {
            $value = $value * 720;  // 720 twips is 0.5 inch
        }

        return parent::setStyleValue($key, $value);
    }

    /**
     * Get style values
     *
     * An experiment to retrieve all style values in one function. This will
     * reduce function call and increase cohesion between functions. Should be
     * implemented in all styles.
     *
     * @ignoreScrutinizerPatch
     * @return array
     */
    public function getStyleValues()
    {
        $styles = array(
            'name'                => $this->getStyleName(),
            'basedOn'             => $this->getBasedOn(),
            'next'                => $this->getNext(),
            'alignment'           => $this->getAlignment(),
            'indentation'         => $this->getIndentation(),
            'spacing'             => $this->getSpace(),
            'pagination'          => array(
                'widowControl'    => $this->hasWidowControl(),
                'keepNext'        => $this->isKeepNext(),
                'keepLines'       => $this->isKeepLines(),
                'pageBreak'       => $this->hasPageBreakBefore(),
            ),
            'numbering'           => array(
                'style'           => $this->getNumStyle(),
                'level'           => $this->getNumLevel(),
            ),
            'tabs'                => $this->getTabs(),
            'shading'             => $this->getShading(),
            'contextualSpacing'   => $this->hasContextualSpacing(),
            'bidi'                => $this->isBidi(),
            'textAlignment'       => $this->getTextAlignment(),
            'suppressAutoHyphens' => $this->hasSuppressAutoHyphens(),
        );

        return $styles;
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    public function getAlignment()
    {
        return $this->alignment;
    }

    /**
     * @since 0.13.0
     *
     * @param string $value
     *
     * @return self
     */
    public function setAlignment($value)
    {
        if (Jc::isValid($value)) {
            $this->alignment = $value;
        }

        return $this;
    }

    /**
     * @deprecated 0.13.0 Use the `getAlignment` method instead.
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getAlign()
    {
        return $this->getAlignment();
    }

    /**
     * @deprecated 0.13.0 Use the `setAlignment` method instead.
     *
     * @param string $value
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setAlign($value = null)
    {
        return $this->setAlignment($value);
    }

    /**
     * Get parent style ID
     *
     * @return string
     */
    public function getBasedOn()
    {
        return $this->basedOn;
    }

    /**
     * Set parent style ID
     *
     * @param string $value
     * @return self
     */
    public function setBasedOn($value = 'Normal')
    {
        $this->basedOn = $value;

        return $this;
    }

    /**
     * Get style for next paragraph
     *
     * @return string
     */
    public function getNext()
    {
        return $this->next;
    }

    /**
     * Set style for next paragraph
     *
     * @param string $value
     * @return self
     */
    public function setNext($value = null)
    {
        $this->next = $value;

        return $this;
    }

    /**
     * Get shading
     *
     * @return \PhpOffice\PhpWord\Style\Indentation
     */
    public function getIndentation()
    {
        return $this->indentation;
    }

    /**
     * Set shading
     *
     * @param mixed $value
     * @return self
     */
    public function setIndentation($value = null)
    {
        $this->setObjectVal($value, 'Indentation', $this->indentation);

        return $this;
    }

    /**
     * Get indentation
     *
     * @return int
     */
    public function getIndent()
    {
        return $this->getChildStyleValue($this->indentation, 'left');
    }

    /**
     * Set indentation
     *
     * @param int $value
     * @return self
     */
    public function setIndent($value = null)
    {
        return $this->setIndentation(array('left' => $value));
    }

    /**
     * Get hanging
     *
     * @return int
     */
    public function getHanging()
    {
        return $this->getChildStyleValue($this->indentation, 'hanging');
    }

    /**
     * Set hanging
     *
     * @param int $value
     * @return self
     */
    public function setHanging($value = null)
    {
        return $this->setIndentation(array('hanging' => $value));
    }

    /**
     * Get spacing
     *
     * @return \PhpOffice\PhpWord\Style\Spacing
     * @todo Rename to getSpacing in 1.0
     */
    public function getSpace()
    {
        return $this->spacing;
    }

    /**
     * Set spacing
     *
     * @param mixed $value
     * @return self
     * @todo Rename to setSpacing in 1.0
     */
    public function setSpace($value = null)
    {
        $this->setObjectVal($value, 'Spacing', $this->spacing);

        return $this;
    }

    /**
     * Get space before paragraph
     *
     * @return int
     */
    public function getSpaceBefore()
    {
        return $this->getChildStyleValue($this->spacing, 'before');
    }

    /**
     * Set space before paragraph
     *
     * @param int $value
     * @return self
     */
    public function setSpaceBefore($value = null)
    {
        return $this->setSpace(array('before' => $value));
    }

    /**
     * Get space after paragraph
     *
     * @return int
     */
    public function getSpaceAfter()
    {
        return $this->getChildStyleValue($this->spacing, 'after');
    }

    /**
     * Set space after paragraph
     *
     * @param int $value
     * @return self
     */
    public function setSpaceAfter($value = null)
    {
        return $this->setSpace(array('after' => $value));
    }

    /**
     * Get spacing between lines
     *
     * @return int|float
     */
    public function getSpacing()
    {
        return $this->getChildStyleValue($this->spacing, 'line');
    }

    /**
     * Set spacing between lines
     *
     * @param int|float $value
     * @return self
     */
    public function setSpacing($value = null)
    {
        return $this->setSpace(array('line' => $value));
    }

    /**
     * Get spacing line rule
     *
     * @return string
     */
    public function getSpacingLineRule()
    {
        return $this->getChildStyleValue($this->spacing, 'lineRule');
    }

    /**
     * Set the spacing line rule
     *
     * @param string $value Possible values are defined in LineSpacingRule
     * @return \PhpOffice\PhpWord\Style\Paragraph
     */
    public function setSpacingLineRule($value)
    {
        return $this->setSpace(array('lineRule' => $value));
    }

    /**
     * Get line height
     *
     * @return int|float
     */
    public function getLineHeight()
    {
        return $this->lineHeight;
    }

    /**
     * Set the line height
     *
     * @param int|float|string $lineHeight
     *
     * @throws \PhpOffice\PhpWord\Exception\InvalidStyleException
     * @return self
     */
    public function setLineHeight($lineHeight)
    {
        if (is_string($lineHeight)) {
            $lineHeight = (float) (preg_replace('/[^0-9\.\,]/', '', $lineHeight));
        }

        if ((!is_int($lineHeight) && !is_float($lineHeight)) || !$lineHeight) {
            throw new InvalidStyleException('Line height must be a valid number');
        }

        $this->lineHeight = $lineHeight;
        $this->setSpacing(($lineHeight - 1) * self::LINE_HEIGHT);
        $this->setSpacingLineRule(\PhpOffice\PhpWord\SimpleType\LineSpacingRule::AUTO);

        return $this;
    }

    /**
     * Get allow first/last line to display on a separate page setting
     *
     * @return bool
     */
    public function hasWidowControl()
    {
        return $this->widowControl;
    }

    /**
     * Set keep paragraph with next paragraph setting
     *
     * @param bool $value
     * @return self
     */
    public function setWidowControl($value = true)
    {
        $this->widowControl = $this->setBoolVal($value, $this->widowControl);

        return $this;
    }

    /**
     * Get keep paragraph with next paragraph setting
     *
     * @return bool
     */
    public function isKeepNext()
    {
        return $this->keepNext;
    }

    /**
     * Set keep paragraph with next paragraph setting
     *
     * @param bool $value
     * @return self
     */
    public function setKeepNext($value = true)
    {
        $this->keepNext = $this->setBoolVal($value, $this->keepNext);

        return $this;
    }

    /**
     * Get keep all lines on one page setting
     *
     * @return bool
     */
    public function isKeepLines()
    {
        return $this->keepLines;
    }

    /**
     * Set keep all lines on one page setting
     *
     * @param bool $value
     * @return self
     */
    public function setKeepLines($value = true)
    {
        $this->keepLines = $this->setBoolVal($value, $this->keepLines);

        return $this;
    }

    /**
     * Get start paragraph on next page setting
     *
     * @return bool
     */
    public function hasPageBreakBefore()
    {
        return $this->pageBreakBefore;
    }

    /**
     * Set start paragraph on next page setting
     *
     * @param bool $value
     * @return self
     */
    public function setPageBreakBefore($value = true)
    {
        $this->pageBreakBefore = $this->setBoolVal($value, $this->pageBreakBefore);

        return $this;
    }

    /**
     * Get numbering style name
     *
     * @return string
     */
    public function getNumStyle()
    {
        return $this->numStyle;
    }

    /**
     * Set numbering style name
     *
     * @param string $value
     * @return self
     */
    public function setNumStyle($value)
    {
        $this->numStyle = $value;

        return $this;
    }

    /**
     * Get numbering level
     *
     * @return int
     */
    public function getNumLevel()
    {
        return $this->numLevel;
    }

    /**
     * Set numbering level
     *
     * @param int $value
     * @return self
     */
    public function setNumLevel($value = 0)
    {
        $this->numLevel = $this->setIntVal($value, $this->numLevel);

        return $this;
    }

    /**
     * Get tabs
     *
     * @return \PhpOffice\PhpWord\Style\Tab[]
     */
    public function getTabs()
    {
        return $this->tabs;
    }

    /**
     * Set tabs
     *
     * @param array $value
     * @return self
     */
    public function setTabs($value = null)
    {
        if (is_array($value)) {
            $this->tabs = $value;
        }

        return $this;
    }

    /**
     * Get allow first/last line to display on a separate page setting
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getWidowControl()
    {
        return $this->hasWidowControl();
    }

    /**
     * Get keep paragraph with next paragraph setting
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getKeepNext()
    {
        return $this->isKeepNext();
    }

    /**
     * Get keep all lines on one page setting
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getKeepLines()
    {
        return $this->isKeepLines();
    }

    /**
     * Get start paragraph on next page setting
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getPageBreakBefore()
    {
        return $this->hasPageBreakBefore();
    }

    /**
     * Get shading
     *
     * @return \PhpOffice\PhpWord\Style\Shading
     */
    public function getShading()
    {
        return $this->shading;
    }

    /**
     * Set shading
     *
     * @param mixed $value
     * @return self
     */
    public function setShading($value = null)
    {
        $this->setObjectVal($value, 'Shading', $this->shading);

        return $this;
    }

    /**
     * Get contextualSpacing
     *
     * @return bool
     */
    public function hasContextualSpacing()
    {
        return $this->contextualSpacing;
    }

    /**
     * Set contextualSpacing
     *
     * @param bool $contextualSpacing
     * @return self
     */
    public function setContextualSpacing($contextualSpacing)
    {
        $this->contextualSpacing = $contextualSpacing;

        return $this;
    }

    /**
     * Get bidirectional
     *
     * @return bool
     */
    public function isBidi()
    {
        return $this->bidi;
    }

    /**
     * Set bidi
     *
     * @param bool $bidi
     *            Set to true to write from right to left
     * @return self
     */
    public function setBidi($bidi)
    {
        $this->bidi = $bidi;

        return $this;
    }

    /**
     * Get textAlignment
     *
     * @return string
     */
    public function getTextAlignment()
    {
        return $this->textAlignment;
    }

    /**
     * Set textAlignment
     *
     * @param string $textAlignment
     * @return self
     */
    public function setTextAlignment($textAlignment)
    {
        TextAlignment::validate($textAlignment);
        $this->textAlignment = $textAlignment;

        return $this;
    }

    /**
     * @return bool
     */
    public function hasSuppressAutoHyphens()
    {
        return $this->suppressAutoHyphens;
    }

    /**
     * @param bool $suppressAutoHyphens
     */
    public function setSuppressAutoHyphens($suppressAutoHyphens)
    {
        $this->suppressAutoHyphens = (bool) $suppressAutoHyphens;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Row.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Table row style
 *
 * @since 0.8.0
 */
class Row extends AbstractStyle
{
    /**
     * Repeat table row on every new page
     *
     * @var bool
     */
    private $tblHeader = false;

    /**
     * Table row cannot break across pages
     *
     * @var bool
     */
    private $cantSplit = false;

    /**
     * Table row exact height
     *
     * @var bool
     */
    private $exactHeight = false;

    /**
     * Create a new row style
     */
    public function __construct()
    {
    }

    /**
     * Is tblHeader
     *
     * @return bool
     */
    public function isTblHeader()
    {
        return $this->tblHeader;
    }

    /**
     * Is tblHeader
     *
     * @param bool $value
     * @return self
     */
    public function setTblHeader($value = true)
    {
        $this->tblHeader = $this->setBoolVal($value, $this->tblHeader);

        return $this;
    }

    /**
     * Is cantSplit
     *
     * @return bool
     */
    public function isCantSplit()
    {
        return $this->cantSplit;
    }

    /**
     * Is cantSplit
     *
     * @param bool $value
     * @return self
     */
    public function setCantSplit($value = true)
    {
        $this->cantSplit = $this->setBoolVal($value, $this->cantSplit);

        return $this;
    }

    /**
     * Is exactHeight
     *
     * @return bool
     */
    public function isExactHeight()
    {
        return $this->exactHeight;
    }

    /**
     * Set exactHeight
     *
     * @param bool $value
     * @return self
     */
    public function setExactHeight($value = true)
    {
        $this->exactHeight = $this->setBoolVal($value, $this->exactHeight);

        return $this;
    }

    /**
     * Get tblHeader
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getTblHeader()
    {
        return $this->isTblHeader();
    }

    /**
     * Get cantSplit
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getCantSplit()
    {
        return $this->isCantSplit();
    }

    /**
     * Get exactHeight
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getExactHeight()
    {
        return $this->isExactHeight();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Section.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\SimpleType\VerticalJc;

/**
 * Section settings
 */
class Section extends Border
{
    /**
     * Page orientation
     *
     * @const string
     */
    const ORIENTATION_PORTRAIT = 'portrait';
    const ORIENTATION_LANDSCAPE = 'landscape';

    /**
     * Page default constants
     *
     * @const int|float
     */
    const DEFAULT_WIDTH = 11905.511811024; // In twips.
    const DEFAULT_HEIGHT = 16837.79527559; // In twips.
    const DEFAULT_MARGIN = 1440;           // In twips.
    const DEFAULT_GUTTER = 0;              // In twips.
    const DEFAULT_HEADER_HEIGHT = 720;     // In twips.
    const DEFAULT_FOOTER_HEIGHT = 720;     // In twips.
    const DEFAULT_COLUMN_COUNT = 1;
    const DEFAULT_COLUMN_SPACING = 720;    // In twips.

    /**
     * Page Orientation
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/a-w_orient-1.html
     */
    private $orientation = self::ORIENTATION_PORTRAIT;

    /**
     * Paper size
     *
     * @var \PhpOffice\PhpWord\Style\Paper
     */
    private $paper;

    /**
     * Page Size Width
     *
     * @var int|float
     */
    private $pageSizeW = self::DEFAULT_WIDTH;

    /**
     * Page Size Height
     *
     * @var int|float
     */
    private $pageSizeH = self::DEFAULT_HEIGHT;

    /**
     * Top margin spacing
     *
     * @var int|float
     */
    private $marginTop = self::DEFAULT_MARGIN;

    /**
     * Left margin spacing
     *
     * @var int|float
     */
    private $marginLeft = self::DEFAULT_MARGIN;

    /**
     * Right margin spacing
     *
     * @var int|float
     */
    private $marginRight = self::DEFAULT_MARGIN;

    /**
     * Bottom margin spacing
     *
     * @var int|float
     */
    private $marginBottom = self::DEFAULT_MARGIN;

    /**
     * Page gutter spacing
     *
     * @var int|float
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_pgMar-1.html
     */
    private $gutter = self::DEFAULT_GUTTER;

    /**
     * Header height
     *
     * @var int|float
     */
    private $headerHeight = self::DEFAULT_HEADER_HEIGHT;

    /**
     * Footer height
     *
     * @var int|float
     */
    private $footerHeight = self::DEFAULT_FOOTER_HEIGHT;

    /**
     * Page Numbering Start
     *
     * @var int
     */
    private $pageNumberingStart;

    /**
     * Section columns count
     *
     * @var int
     */
    private $colsNum = self::DEFAULT_COLUMN_COUNT;

    /**
     * Section spacing between columns
     *
     * @var int|float
     */
    private $colsSpace = self::DEFAULT_COLUMN_SPACING;

    /**
     * Section break type
     *
     * Options:
     * - nextPage: Next page section break
     * - nextColumn: Column section break
     * - continuous: Continuous section break
     * - evenPage: Even page section break
     * - oddPage: Odd page section break
     *
     * @var string
     */
    private $breakType;

    /**
     * Line numbering
     *
     * @var \PhpOffice\PhpWord\Style\LineNumbering
     * @see  http://www.schemacentral.com/sc/ooxml/e-w_lnNumType-1.html
     */
    private $lineNumbering;

    /**
     * Vertical Text Alignment on Page
     * One of \PhpOffice\PhpWord\SimpleType\VerticalJc
     *
     * @var string
     */
    private $vAlign;

    /**
     * Create new instance
     */
    public function __construct()
    {
        $this->setPaperSize();
    }

    /**
     * Get paper size
     *
     * @return string
     */
    public function getPaperSize()
    {
        return $this->paper->getSize();
    }

    /**
     * Set paper size
     *
     * @param string $value
     * @return self
     */
    public function setPaperSize($value = '')
    {
        if (!$value) {
            $value = Settings::getDefaultPaper();
        }
        if ($this->paper === null) {
            $this->paper = new Paper();
        }
        $this->paper->setSize($value);
        $this->pageSizeW = $this->paper->getWidth();
        $this->pageSizeH = $this->paper->getHeight();

        return $this;
    }

    /**
     * Set Setting Value
     *
     * @param string $key
     * @param string $value
     * @return self
     */
    public function setSettingValue($key, $value)
    {
        return $this->setStyleValue($key, $value);
    }

    /**
     * Set orientation
     *
     * @param string $value
     * @return self
     */
    public function setOrientation($value = null)
    {
        $enum = array(self::ORIENTATION_PORTRAIT, self::ORIENTATION_LANDSCAPE);
        $this->orientation = $this->setEnumVal($value, $enum, $this->orientation);

        /** @var int|float $longSide Type hint */
        $longSide = $this->pageSizeW >= $this->pageSizeH ? $this->pageSizeW : $this->pageSizeH;

        /** @var int|float $shortSide Type hint */
        $shortSide = $this->pageSizeW < $this->pageSizeH ? $this->pageSizeW : $this->pageSizeH;

        if ($this->orientation == self::ORIENTATION_PORTRAIT) {
            $this->pageSizeW = $shortSide;
            $this->pageSizeH = $longSide;
        } else {
            $this->pageSizeW = $longSide;
            $this->pageSizeH = $shortSide;
        }

        return $this;
    }

    /**
     * Get Page Orientation
     *
     * @return string
     */
    public function getOrientation()
    {
        return $this->orientation;
    }

    /**
     * Set Portrait Orientation
     *
     * @return self
     */
    public function setPortrait()
    {
        return $this->setOrientation(self::ORIENTATION_PORTRAIT);
    }

    /**
     * Set Landscape Orientation
     *
     * @return self
     */
    public function setLandscape()
    {
        return $this->setOrientation(self::ORIENTATION_LANDSCAPE);
    }

    /**
     * Get Page Size Width
     *
     * @return int|float|null
     *
     * @since 0.12.0
     */
    public function getPageSizeW()
    {
        return $this->pageSizeW;
    }

    /**
     * @param int|float|null $value
     *
     * @return \PhpOffice\PhpWord\Style\Section
     *
     * @since 0.12.0
     */
    public function setPageSizeW($value = null)
    {
        $this->pageSizeW = $this->setNumericVal($value, self::DEFAULT_WIDTH);

        return $this;
    }

    /**
     * Get Page Size Height
     *
     * @return int|float|null
     *
     * @since 0.12.0
     */
    public function getPageSizeH()
    {
        return $this->pageSizeH;
    }

    /**
     * @param int|float|null $value
     *
     * @return \PhpOffice\PhpWord\Style\Section
     *
     * @since 0.12.0
     */
    public function setPageSizeH($value = null)
    {
        $this->pageSizeH = $this->setNumericVal($value, self::DEFAULT_HEIGHT);

        return $this;
    }

    /**
     * Get Margin Top
     *
     * @return int|float
     */
    public function getMarginTop()
    {
        return $this->marginTop;
    }

    /**
     * Set Margin Top
     *
     * @param int|float $value
     * @return self
     */
    public function setMarginTop($value = null)
    {
        $this->marginTop = $this->setNumericVal($value, self::DEFAULT_MARGIN);

        return $this;
    }

    /**
     * Get Margin Left
     *
     * @return int|float
     */
    public function getMarginLeft()
    {
        return $this->marginLeft;
    }

    /**
     * Set Margin Left
     *
     * @param int|float $value
     * @return self
     */
    public function setMarginLeft($value = null)
    {
        $this->marginLeft = $this->setNumericVal($value, self::DEFAULT_MARGIN);

        return $this;
    }

    /**
     * Get Margin Right
     *
     * @return int|float
     */
    public function getMarginRight()
    {
        return $this->marginRight;
    }

    /**
     * Set Margin Right
     *
     * @param int|float $value
     * @return self
     */
    public function setMarginRight($value = null)
    {
        $this->marginRight = $this->setNumericVal($value, self::DEFAULT_MARGIN);

        return $this;
    }

    /**
     * Get Margin Bottom
     *
     * @return int|float
     */
    public function getMarginBottom()
    {
        return $this->marginBottom;
    }

    /**
     * Set Margin Bottom
     *
     * @param int|float $value
     * @return self
     */
    public function setMarginBottom($value = null)
    {
        $this->marginBottom = $this->setNumericVal($value, self::DEFAULT_MARGIN);

        return $this;
    }

    /**
     * Get gutter
     *
     * @return int|float
     */
    public function getGutter()
    {
        return $this->gutter;
    }

    /**
     * Set gutter
     *
     * @param int|float $value
     * @return self
     */
    public function setGutter($value = null)
    {
        $this->gutter = $this->setNumericVal($value, self::DEFAULT_GUTTER);

        return $this;
    }

    /**
     * Get Header Height
     *
     * @return int|float
     */
    public function getHeaderHeight()
    {
        return $this->headerHeight;
    }

    /**
     * Set Header Height
     *
     * @param int|float $value
     * @return self
     */
    public function setHeaderHeight($value = null)
    {
        $this->headerHeight = $this->setNumericVal($value, self::DEFAULT_HEADER_HEIGHT);

        return $this;
    }

    /**
     * Get Footer Height
     *
     * @return int|float
     */
    public function getFooterHeight()
    {
        return $this->footerHeight;
    }

    /**
     * Set Footer Height
     *
     * @param int|float $value
     * @return self
     */
    public function setFooterHeight($value = null)
    {
        $this->footerHeight = $this->setNumericVal($value, self::DEFAULT_FOOTER_HEIGHT);

        return $this;
    }

    /**
     * Get page numbering start
     *
     * @return null|int
     */
    public function getPageNumberingStart()
    {
        return $this->pageNumberingStart;
    }

    /**
     * Set page numbering start
     *
     * @param null|int $pageNumberingStart
     * @return self
     */
    public function setPageNumberingStart($pageNumberingStart = null)
    {
        $this->pageNumberingStart = $pageNumberingStart;

        return $this;
    }

    /**
     * Get Section Columns Count
     *
     * @return int
     */
    public function getColsNum()
    {
        return $this->colsNum;
    }

    /**
     * Set Section Columns Count
     *
     * @param int $value
     * @return self
     */
    public function setColsNum($value = null)
    {
        $this->colsNum = $this->setIntVal($value, self::DEFAULT_COLUMN_COUNT);

        return $this;
    }

    /**
     * Get Section Space Between Columns
     *
     * @return int|float
     */
    public function getColsSpace()
    {
        return $this->colsSpace;
    }

    /**
     * Set Section Space Between Columns
     *
     * @param int|float $value
     * @return self
     */
    public function setColsSpace($value = null)
    {
        $this->colsSpace = $this->setNumericVal($value, self::DEFAULT_COLUMN_SPACING);

        return $this;
    }

    /**
     * Get Break Type
     *
     * @return string
     */
    public function getBreakType()
    {
        return $this->breakType;
    }

    /**
     * Set Break Type
     *
     * @param string $value
     * @return self
     */
    public function setBreakType($value = null)
    {
        $this->breakType = $value;

        return $this;
    }

    /**
     * Get line numbering
     *
     * @return \PhpOffice\PhpWord\Style\LineNumbering
     */
    public function getLineNumbering()
    {
        return $this->lineNumbering;
    }

    /**
     * Set line numbering
     *
     * @param mixed $value
     * @return self
     */
    public function setLineNumbering($value = null)
    {
        $this->setObjectVal($value, 'LineNumbering', $this->lineNumbering);

        return $this;
    }

    /**
     * Get vertical alignment
     *
     * @return string
     */
    public function getVAlign()
    {
        return $this->vAlign;
    }

    /**
     * Set vertical alignment
     *
     * @param string $value
     * @return self
     */
    public function setVAlign($value = null)
    {
        VerticalJc::validate($value);
        $this->vAlign = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Shading.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Shading style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_Shd.html
 * @since 0.10.0
 */
class Shading extends AbstractStyle
{
    /**
     * Pattern constants (partly)
     *
     * @const string
     * @see  http://www.schemacentral.com/sc/ooxml/t-w_ST_Shd.html
     */
    const PATTERN_CLEAR = 'clear'; // No pattern
    const PATTERN_SOLID = 'solid'; // 100% fill pattern
    const PATTERN_HSTRIPE = 'horzStripe'; // Horizontal stripe pattern
    const PATTERN_VSTRIPE = 'vertStripe'; // Vertical stripe pattern
    const PATTERN_DSTRIPE = 'diagStripe'; // Diagonal stripe pattern
    const PATTERN_HCROSS = 'horzCross'; // Horizontal cross pattern
    const PATTERN_DCROSS = 'diagCross'; // Diagonal cross pattern

    /**
     * Shading pattern
     *
     * @var string
     * @see  http://www.schemacentral.com/sc/ooxml/t-w_ST_Shd.html
     */
    private $pattern = self::PATTERN_CLEAR;

    /**
     * Shading pattern color
     *
     * @var string
     */
    private $color;

    /**
     * Shading background color
     *
     * @var string
     */
    private $fill;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get pattern
     *
     * @return string
     */
    public function getPattern()
    {
        return $this->pattern;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setPattern($value = null)
    {
        $enum = array(
            self::PATTERN_CLEAR, self::PATTERN_SOLID, self::PATTERN_HSTRIPE,
            self::PATTERN_VSTRIPE, self::PATTERN_DSTRIPE, self::PATTERN_HCROSS, self::PATTERN_DCROSS,
        );
        $this->pattern = $this->setEnumVal($value, $enum, $this->pattern);

        return $this;
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set pattern
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }

    /**
     * Get fill
     *
     * @return string
     */
    public function getFill()
    {
        return $this->fill;
    }

    /**
     * Set fill
     *
     * @param string $value
     * @return self
     */
    public function setFill($value = null)
    {
        $this->fill = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Shadow.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Shadow style
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-v_CT_Shadow.html
 * @since 0.12.0
 */
class Shadow extends AbstractStyle
{
    /**
     * Color
     *
     * @var string
     */
    private $color;

    /**
     * Offset; Format: 3pt,3pt
     *
     * @var string
     */
    private $offset;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get color
     *
     * @return string
     */
    public function getColor()
    {
        return $this->color;
    }

    /**
     * Set color
     *
     * @param string $value
     * @return self
     */
    public function setColor($value = null)
    {
        $this->color = $value;

        return $this;
    }

    /**
     * Get offset
     *
     * @return string
     */
    public function getOffset()
    {
        return $this->offset;
    }

    /**
     * Set offset
     *
     * @param string $value
     * @return self
     */
    public function setOffset($value = null)
    {
        $this->offset = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Shape.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Shape style
 *
 * @since 0.12.0
 * @todo Skew http://www.schemacentral.com/sc/ooxml/t-o_CT_Skew.html
 */
class Shape extends AbstractStyle
{
    /**
     * Points
     *
     * - Arc: startAngle endAngle; 0 = top center, moving clockwise
     * - Curve: from-x1,from-y1 to-x2,to-y2 control1-x,control1-y control2-x,control2-y
     * - Line: from-x1,from-y1 to-x2,to-y2
     * - Polyline: x1,y1 x2,y2 ...
     * - Rect and oval: Not applicable
     *
     * @var string
     */
    private $points;

    /**
     * Roundness measure of corners; 0 = straightest (rectangular); 1 = roundest (circle/oval)
     *
     * Only for rect
     *
     * @var int|float
     */
    private $roundness;

    /**
     * Frame
     *
     * @var \PhpOffice\PhpWord\Style\Frame
     */
    private $frame;

    /**
     * Fill
     *
     * @var \PhpOffice\PhpWord\Style\Fill
     */
    private $fill;

    /**
     * Outline
     *
     * @var \PhpOffice\PhpWord\Style\Outline
     */
    private $outline;

    /**
     * Shadow
     *
     * @var \PhpOffice\PhpWord\Style\Shadow
     */
    private $shadow;

    /**
     * 3D extrusion
     *
     * @var \PhpOffice\PhpWord\Style\Extrusion
     */
    private $extrusion;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get points
     *
     * @return string
     */
    public function getPoints()
    {
        return $this->points;
    }

    /**
     * Set points
     *
     * @param string $value
     * @return self
     */
    public function setPoints($value = null)
    {
        $this->points = $value;

        return $this;
    }

    /**
     * Get roundness
     *
     * @return int|float
     */
    public function getRoundness()
    {
        return $this->roundness;
    }

    /**
     * Set roundness
     *
     * @param int|float $value
     * @return self
     */
    public function setRoundness($value = null)
    {
        $this->roundness = $this->setNumericVal($value, null);

        return $this;
    }

    /**
     * Get frame
     *
     * @return \PhpOffice\PhpWord\Style\Frame
     */
    public function getFrame()
    {
        return $this->frame;
    }

    /**
     * Set frame
     *
     * @param mixed $value
     * @return self
     */
    public function setFrame($value = null)
    {
        $this->setObjectVal($value, 'Frame', $this->frame);

        return $this;
    }

    /**
     * Get fill
     *
     * @return \PhpOffice\PhpWord\Style\Fill
     */
    public function getFill()
    {
        return $this->fill;
    }

    /**
     * Set fill
     *
     * @param mixed $value
     * @return self
     */
    public function setFill($value = null)
    {
        $this->setObjectVal($value, 'Fill', $this->fill);

        return $this;
    }

    /**
     * Get outline
     *
     * @return \PhpOffice\PhpWord\Style\Outline
     */
    public function getOutline()
    {
        return $this->outline;
    }

    /**
     * Set outline
     *
     * @param mixed $value
     * @return self
     */
    public function setOutline($value = null)
    {
        $this->setObjectVal($value, 'Outline', $this->outline);

        return $this;
    }

    /**
     * Get shadow
     *
     * @return \PhpOffice\PhpWord\Style\Shadow
     */
    public function getShadow()
    {
        return $this->shadow;
    }

    /**
     * Set shadow
     *
     * @param mixed $value
     * @return self
     */
    public function setShadow($value = null)
    {
        $this->setObjectVal($value, 'Shadow', $this->shadow);

        return $this;
    }

    /**
     * Get 3D extrusion
     *
     * @return \PhpOffice\PhpWord\Style\Extrusion
     */
    public function getExtrusion()
    {
        return $this->extrusion;
    }

    /**
     * Set 3D extrusion
     *
     * @param mixed $value
     * @return self
     */
    public function setExtrusion($value = null)
    {
        $this->setObjectVal($value, 'Extrusion', $this->extrusion);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Spacing.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\SimpleType\LineSpacingRule;

/**
 * Spacing between lines and above/below paragraph style
 *
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_Spacing.html
 * @since 0.10.0
 */
class Spacing extends AbstractStyle
{
    /**
     * Spacing above paragraph (twip)
     *
     * @var int|float
     */
    private $before;

    /**
     * Spacing below paragraph (twip)
     *
     * @var int|float
     */
    private $after;

    /**
     * Spacing between lines in paragraph (twip)
     *
     * @var int|float
     */
    private $line;

    /**
     * Type of spacing between lines
     *
     * @var string
     */
    private $lineRule = LineSpacingRule::AUTO;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get before
     *
     * @return int|float
     */
    public function getBefore()
    {
        return $this->before;
    }

    /**
     * Set before
     *
     * @param int|float $value
     * @return self
     */
    public function setBefore($value = null)
    {
        $this->before = $this->setNumericVal($value, $this->before);

        return $this;
    }

    /**
     * Get after
     *
     * @return int|float
     */
    public function getAfter()
    {
        return $this->after;
    }

    /**
     * Set after
     *
     * @param int|float $value
     * @return self
     */
    public function setAfter($value = null)
    {
        $this->after = $this->setNumericVal($value, $this->after);

        return $this;
    }

    /**
     * Get line
     *
     * @return int|float
     */
    public function getLine()
    {
        return $this->line;
    }

    /**
     * Set distance
     *
     * @param int|float $value
     * @return self
     */
    public function setLine($value = null)
    {
        $this->line = $this->setNumericVal($value, $this->line);

        return $this;
    }

    /**
     * Get line rule
     *
     * @return string
     */
    public function getLineRule()
    {
        return $this->lineRule;
    }

    /**
     * Set line rule
     *
     * @param string $value
     * @return self
     */
    public function setLineRule($value = null)
    {
        LineSpacingRule::validate($value);
        $this->lineRule = $value;

        return $this;
    }

    /**
     * Get line rule
     *
     * @return string
     * @deprecated Use getLineRule() instead
     * @codeCoverageIgnore
     */
    public function getRule()
    {
        return $this->lineRule;
    }

    /**
     * Set line rule
     *
     * @param string $value
     * @return self
     * @deprecated Use setLineRule() instead
     * @codeCoverageIgnore
     */
    public function setRule($value = null)
    {
        $this->lineRule = $value;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Tab.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * Tab style
 */
class Tab extends AbstractStyle
{
    /**
     * Tab stop types
     *
     * @const string
     */
    const TAB_STOP_CLEAR = 'clear';
    const TAB_STOP_LEFT = 'left';
    const TAB_STOP_CENTER = 'center';
    const TAB_STOP_RIGHT = 'right';
    const TAB_STOP_DECIMAL = 'decimal';
    const TAB_STOP_BAR = 'bar';
    const TAB_STOP_NUM = 'num';

    /**
     * Tab leader types
     *
     * @const string
     */
    const TAB_LEADER_NONE = 'none';
    const TAB_LEADER_DOT = 'dot';
    const TAB_LEADER_HYPHEN = 'hyphen';
    const TAB_LEADER_UNDERSCORE = 'underscore';
    const TAB_LEADER_HEAVY = 'heavy';
    const TAB_LEADER_MIDDLEDOT = 'middleDot';

    /**
     * Tab stop type
     *
     * @var string
     */
    private $type = self::TAB_STOP_CLEAR;

    /**
     * Tab leader character
     *
     * @var string
     */
    private $leader = self::TAB_LEADER_NONE;

    /**
     * Tab stop position (twip)
     *
     * @var int|float
     */
    private $position = 0;

    /**
     * Create a new instance of Tab. Both $type and $leader
     * must conform to the values put forth in the schema. If they do not
     * they will be changed to default values.
     *
     * @param string $type Defaults to 'clear' if value is not possible
     * @param int $position Must be numeric; otherwise defaults to 0
     * @param string $leader Defaults to null if value is not possible
     */
    public function __construct($type = null, $position = 0, $leader = null)
    {
        $stopTypes = array(
            self::TAB_STOP_CLEAR, self::TAB_STOP_LEFT, self::TAB_STOP_CENTER,
            self::TAB_STOP_RIGHT, self::TAB_STOP_DECIMAL, self::TAB_STOP_BAR, self::TAB_STOP_NUM,
        );
        $leaderTypes = array(
            self::TAB_LEADER_NONE, self::TAB_LEADER_DOT, self::TAB_LEADER_HYPHEN,
            self::TAB_LEADER_UNDERSCORE, self::TAB_LEADER_HEAVY, self::TAB_LEADER_MIDDLEDOT,
        );

        $this->type = $this->setEnumVal($type, $stopTypes, $this->type);
        $this->position = $this->setNumericVal($position, $this->position);
        $this->leader = $this->setEnumVal($leader, $leaderTypes, $this->leader);
    }

    /**
     * Get stop type
     *
     * @return string
     */
    public function getType()
    {
        return $this->type;
    }

    /**
     * Set stop type
     *
     * @param string $value
     * @return self
     */
    public function setType($value)
    {
        $enum = array(
            self::TAB_STOP_CLEAR, self::TAB_STOP_LEFT, self::TAB_STOP_CENTER,
            self::TAB_STOP_RIGHT, self::TAB_STOP_DECIMAL, self::TAB_STOP_BAR,
            self::TAB_STOP_NUM,
        );
        $this->type = $this->setEnumVal($value, $enum, $this->type);

        return $this;
    }

    /**
     * Get leader
     *
     * @return string
     */
    public function getLeader()
    {
        return $this->leader;
    }

    /**
     * Set leader
     *
     * @param string $value
     * @return self
     */
    public function setLeader($value)
    {
        $enum = array(
            self::TAB_LEADER_NONE, self::TAB_LEADER_DOT, self::TAB_LEADER_HYPHEN,
            self::TAB_LEADER_UNDERSCORE, self::TAB_LEADER_HEAVY, self::TAB_LEADER_MIDDLEDOT,
        );
        $this->leader = $this->setEnumVal($value, $enum, $this->leader);

        return $this;
    }

    /**
     * Get position
     *
     * @return int|float
     */
    public function getPosition()
    {
        return $this->position;
    }

    /**
     * Set position
     *
     * @param int|float $value
     * @return self
     */
    public function setPosition($value)
    {
        $this->position = $this->setNumericVal($value, $this->position);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

use PhpOffice\PhpWord\ComplexType\TblWidth as TblWidthComplexType;
use PhpOffice\PhpWord\SimpleType\Jc;
use PhpOffice\PhpWord\SimpleType\JcTable;
use PhpOffice\PhpWord\SimpleType\TblWidth;

class Table extends Border
{
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\TblWidth::AUTO instead
     */
    const WIDTH_AUTO = 'auto'; // Automatically determined width
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\TblWidth::PERCENT instead
     */
    const WIDTH_PERCENT = 'pct'; // Width in fiftieths (1/50) of a percent (1% = 50 unit)
    /**
     * @deprecated Use \PhpOffice\PhpWord\SimpleType\TblWidth::TWIP instead
     */
    const WIDTH_TWIP = 'dxa'; // Width in twentieths (1/20) of a point (twip)

    //values for http://www.datypic.com/sc/ooxml/t-w_ST_TblLayoutType.html
    /**
     * AutoFit Table Layout
     *
     * @var string
     */
    const LAYOUT_AUTO = 'autofit';
    /**
     * Fixed Width Table Layout
     *
     * @var string
     */
    const LAYOUT_FIXED = 'fixed';

    /**
     * Is this a first row style?
     *
     * @var bool
     */
    private $isFirstRow = false;

    /**
     * Style for first row
     *
     * @var \PhpOffice\PhpWord\Style\Table
     */
    private $firstRowStyle;

    /**
     * Cell margin top
     *
     * @var int
     */
    private $cellMarginTop;

    /**
     * Cell margin left
     *
     * @var int
     */
    private $cellMarginLeft;

    /**
     * Cell margin right
     *
     * @var int
     */
    private $cellMarginRight;

    /**
     * Cell margin bottom
     *
     * @var int
     */
    private $cellMarginBottom;

    /**
     * Border size inside horizontal
     *
     * @var int
     */
    private $borderInsideHSize;

    /**
     * Border color inside horizontal
     *
     * @var string
     */
    private $borderInsideHColor;

    /**
     * Border size inside vertical
     *
     * @var int
     */
    private $borderInsideVSize;

    /**
     * Border color inside vertical
     *
     * @var string
     */
    private $borderInsideVColor;

    /**
     * Shading
     *
     * @var \PhpOffice\PhpWord\Style\Shading
     */
    private $shading;

    /**
     * @var string
     */
    private $alignment = '';

    /**
     * @var int|float Width value
     */
    private $width = 0;

    /**
     * @var string Width unit
     */
    private $unit = TblWidth::AUTO;

    /**
     * @var int|float cell spacing value
     */
    protected $cellSpacing = null;

    /**
     * @var string Table Layout
     */
    private $layout = self::LAYOUT_AUTO;

    /**
     * Position
     *
     * @var \PhpOffice\PhpWord\Style\TablePosition
     */
    private $position;

    /** @var TblWidthComplexType|null */
    private $indent;

    /**
     * The width of each column, computed based on the max cell width of each column
     *
     * @var int[]
     */
    private $columnWidths;

    /**
     * Visually Right to Left Table
     *
     * @see  http://www.datypic.com/sc/ooxml/e-w_bidiVisual-1.html
     * @var bool
     */
    private $bidiVisual = false;

    /**
     * Create new table style
     *
     * @param mixed $tableStyle
     * @param mixed $firstRowStyle
     */
    public function __construct($tableStyle = null, $firstRowStyle = null)
    {
        // Clone first row from table style, but with certain properties disabled
        if ($firstRowStyle !== null && is_array($firstRowStyle)) {
            $this->firstRowStyle = clone $this;
            $this->firstRowStyle->isFirstRow = true;
            unset($this->firstRowStyle->firstRowStyle, $this->firstRowStyle->borderInsideHSize, $this->firstRowStyle->borderInsideHColor, $this->firstRowStyle->borderInsideVSize, $this->firstRowStyle->borderInsideVColor, $this->firstRowStyle->cellMarginTop, $this->firstRowStyle->cellMarginLeft, $this->firstRowStyle->cellMarginRight, $this->firstRowStyle->cellMarginBottom, $this->firstRowStyle->cellSpacing);
            $this->firstRowStyle->setStyleByArray($firstRowStyle);
        }

        if ($tableStyle !== null && is_array($tableStyle)) {
            $this->setStyleByArray($tableStyle);
        }
    }

    /**
     * @param float|int $cellSpacing
     */
    public function setCellSpacing($cellSpacing = null)
    {
        $this->cellSpacing = $cellSpacing;
    }

    /**
     * @return float|int
     */
    public function getCellSpacing()
    {
        return $this->cellSpacing;
    }

    /**
     * Set first row
     *
     * @return \PhpOffice\PhpWord\Style\Table
     */
    public function getFirstRow()
    {
        return $this->firstRowStyle;
    }

    /**
     * Get background
     *
     * @return string
     */
    public function getBgColor()
    {
        if ($this->shading !== null) {
            return $this->shading->getFill();
        }

        return null;
    }

    /**
     * Set background
     *
     * @param string $value
     * @return self
     */
    public function setBgColor($value = null)
    {
        $this->setShading(array('fill' => $value));

        return $this;
    }

    /**
     * Get TLRBHV Border Size
     *
     * @return int[]
     */
    public function getBorderSize()
    {
        return array(
            $this->getBorderTopSize(),
            $this->getBorderLeftSize(),
            $this->getBorderRightSize(),
            $this->getBorderBottomSize(),
            $this->getBorderInsideHSize(),
            $this->getBorderInsideVSize(),
        );
    }

    /**
     * Set TLRBHV Border Size
     *
     * @param int $value Border size in eighths of a point (1/8 point)
     * @return self
     */
    public function setBorderSize($value = null)
    {
        $this->setBorderTopSize($value);
        $this->setBorderLeftSize($value);
        $this->setBorderRightSize($value);
        $this->setBorderBottomSize($value);
        $this->setBorderInsideHSize($value);
        $this->setBorderInsideVSize($value);

        return $this;
    }

    /**
     * Get TLRBHV Border Color
     *
     * @return string[]
     */
    public function getBorderColor()
    {
        return array(
            $this->getBorderTopColor(),
            $this->getBorderLeftColor(),
            $this->getBorderRightColor(),
            $this->getBorderBottomColor(),
            $this->getBorderInsideHColor(),
            $this->getBorderInsideVColor(),
        );
    }

    /**
     * Set TLRBHV Border Color
     *
     * @param string $value
     * @return self
     */
    public function setBorderColor($value = null)
    {
        $this->setBorderTopColor($value);
        $this->setBorderLeftColor($value);
        $this->setBorderRightColor($value);
        $this->setBorderBottomColor($value);
        $this->setBorderInsideHColor($value);
        $this->setBorderInsideVColor($value);

        return $this;
    }

    /**
     * Get border size inside horizontal
     *
     * @return int
     */
    public function getBorderInsideHSize()
    {
        return $this->getTableOnlyProperty('borderInsideHSize');
    }

    /**
     * Set border size inside horizontal
     *
     * @param int $value
     * @return self
     */
    public function setBorderInsideHSize($value = null)
    {
        return $this->setTableOnlyProperty('borderInsideHSize', $value);
    }

    /**
     * Get border color inside horizontal
     *
     * @return string
     */
    public function getBorderInsideHColor()
    {
        return $this->getTableOnlyProperty('borderInsideHColor');
    }

    /**
     * Set border color inside horizontal
     *
     * @param string $value
     * @return self
     */
    public function setBorderInsideHColor($value = null)
    {
        return $this->setTableOnlyProperty('borderInsideHColor', $value, false);
    }

    /**
     * Get border size inside vertical
     *
     * @return int
     */
    public function getBorderInsideVSize()
    {
        return $this->getTableOnlyProperty('borderInsideVSize');
    }

    /**
     * Set border size inside vertical
     *
     * @param int $value
     * @return self
     */
    public function setBorderInsideVSize($value = null)
    {
        return $this->setTableOnlyProperty('borderInsideVSize', $value);
    }

    /**
     * Get border color inside vertical
     *
     * @return string
     */
    public function getBorderInsideVColor()
    {
        return $this->getTableOnlyProperty('borderInsideVColor');
    }

    /**
     * Set border color inside vertical
     *
     * @param string $value
     * @return self
     */
    public function setBorderInsideVColor($value = null)
    {
        return $this->setTableOnlyProperty('borderInsideVColor', $value, false);
    }

    /**
     * Get cell margin top
     *
     * @return int
     */
    public function getCellMarginTop()
    {
        return $this->getTableOnlyProperty('cellMarginTop');
    }

    /**
     * Set cell margin top
     *
     * @param int $value
     * @return self
     */
    public function setCellMarginTop($value = null)
    {
        return $this->setTableOnlyProperty('cellMarginTop', $value);
    }

    /**
     * Get cell margin left
     *
     * @return int
     */
    public function getCellMarginLeft()
    {
        return $this->getTableOnlyProperty('cellMarginLeft');
    }

    /**
     * Set cell margin left
     *
     * @param int $value
     * @return self
     */
    public function setCellMarginLeft($value = null)
    {
        return $this->setTableOnlyProperty('cellMarginLeft', $value);
    }

    /**
     * Get cell margin right
     *
     * @return int
     */
    public function getCellMarginRight()
    {
        return $this->getTableOnlyProperty('cellMarginRight');
    }

    /**
     * Set cell margin right
     *
     * @param int $value
     * @return self
     */
    public function setCellMarginRight($value = null)
    {
        return $this->setTableOnlyProperty('cellMarginRight', $value);
    }

    /**
     * Get cell margin bottom
     *
     * @return int
     */
    public function getCellMarginBottom()
    {
        return $this->getTableOnlyProperty('cellMarginBottom');
    }

    /**
     * Set cell margin bottom
     *
     * @param int $value
     * @return self
     */
    public function setCellMarginBottom($value = null)
    {
        return $this->setTableOnlyProperty('cellMarginBottom', $value);
    }

    /**
     * Get cell margin
     *
     * @return int[]
     */
    public function getCellMargin()
    {
        return array(
            $this->cellMarginTop,
            $this->cellMarginLeft,
            $this->cellMarginRight,
            $this->cellMarginBottom,
        );
    }

    /**
     * Set TLRB cell margin
     *
     * @param int $value Margin in twips
     * @return self
     */
    public function setCellMargin($value = null)
    {
        $this->setCellMarginTop($value);
        $this->setCellMarginLeft($value);
        $this->setCellMarginRight($value);
        $this->setCellMarginBottom($value);

        return $this;
    }

    /**
     * Check if any of the margin is not null
     *
     * @return bool
     */
    public function hasMargin()
    {
        $margins = $this->getCellMargin();

        return $margins !== array_filter($margins, 'is_null');
    }

    /**
     * Get shading
     *
     * @return \PhpOffice\PhpWord\Style\Shading
     */
    public function getShading()
    {
        return $this->shading;
    }

    /**
     * Set shading
     *
     * @param mixed $value
     * @return self
     */
    public function setShading($value = null)
    {
        $this->setObjectVal($value, 'Shading', $this->shading);

        return $this;
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    public function getAlignment()
    {
        return $this->alignment;
    }

    /**
     * @since 0.13.0
     *
     * @param string $value
     *
     * @return self
     */
    public function setAlignment($value)
    {
        if (JcTable::isValid($value) || Jc::isValid($value)) {
            $this->alignment = $value;
        }

        return $this;
    }

    /**
     * @deprecated 0.13.0 Use the `getAlignment` method instead.
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function getAlign()
    {
        return $this->getAlignment();
    }

    /**
     * @deprecated 0.13.0 Use the `setAlignment` method instead.
     *
     * @param string $value
     *
     * @return self
     *
     * @codeCoverageIgnore
     */
    public function setAlign($value = null)
    {
        return $this->setAlignment($value);
    }

    /**
     * Get width
     *
     * @return int|float
     */
    public function getWidth()
    {
        return $this->width;
    }

    /**
     * Set width
     *
     * @param int|float $value
     * @return self
     */
    public function setWidth($value = null)
    {
        $this->width = $this->setNumericVal($value, $this->width);

        return $this;
    }

    /**
     * Get width unit
     *
     * @return string
     */
    public function getUnit()
    {
        return $this->unit;
    }

    /**
     * Set width unit
     *
     * @param string $value
     * @return self
     */
    public function setUnit($value = null)
    {
        TblWidth::validate($value);
        $this->unit = $value;

        return $this;
    }

    /**
     * Get layout
     *
     * @return string
     */
    public function getLayout()
    {
        return $this->layout;
    }

    /**
     * Set layout
     *
     * @param string $value
     * @return self
     */
    public function setLayout($value = null)
    {
        $enum = array(self::LAYOUT_AUTO, self::LAYOUT_FIXED);
        $this->layout = $this->setEnumVal($value, $enum, $this->layout);

        return $this;
    }

    /**
     * Get table style only property by checking if it's a firstRow
     *
     * This is necessary since firstRow style is cloned from table style but
     * without certain properties activated, e.g. margins
     *
     * @param string $property
     * @return int|string|null
     */
    private function getTableOnlyProperty($property)
    {
        if (false === $this->isFirstRow) {
            return $this->$property;
        }

        return null;
    }

    /**
     * Set table style only property by checking if it's a firstRow
     *
     * This is necessary since firstRow style is cloned from table style but
     * without certain properties activated, e.g. margins
     *
     * @param string $property
     * @param int|string $value
     * @param bool $isNumeric
     * @return self
     */
    private function setTableOnlyProperty($property, $value, $isNumeric = true)
    {
        if (false === $this->isFirstRow) {
            if (true === $isNumeric) {
                $this->$property = $this->setNumericVal($value, $this->$property);
            } else {
                $this->$property = $value;
            }
        }

        return $this;
    }

    /**
     * Get position
     *
     * @return \PhpOffice\PhpWord\Style\TablePosition
     */
    public function getPosition()
    {
        return $this->position;
    }

    /**
     * Set position
     *
     * @param mixed $value
     * @return self
     */
    public function setPosition($value = null)
    {
        $this->setObjectVal($value, 'TablePosition', $this->position);

        return $this;
    }

    /**
     * @return TblWidthComplexType
     */
    public function getIndent()
    {
        return $this->indent;
    }

    /**
     * @param TblWidthComplexType $indent
     * @return self
     * @see http://www.datypic.com/sc/ooxml/e-w_tblInd-1.html
     */
    public function setIndent(TblWidthComplexType $indent)
    {
        $this->indent = $indent;

        return $this;
    }

    /**
     * Get the columnWidths
     *
     * @return null|int[]
     */
    public function getColumnWidths()
    {
        return $this->columnWidths;
    }

    /**
     * The column widths
     *
     * @param int[] $value
     */
    public function setColumnWidths(array $value = null)
    {
        $this->columnWidths = $value;
    }

    /**
     * Get bidiVisual
     *
     * @return bool
     */
    public function isBidiVisual()
    {
        return $this->bidiVisual;
    }

    /**
     * Set bidiVisual
     *
     * @param bool $bidi
     *            Set to true to visually present table as Right to Left
     * @return self
     */
    public function setBidiVisual($bidi)
    {
        $this->bidiVisual = $bidi;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/TablePosition.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * TablePosition style
 *
 * @see http://www.datypic.com/sc/ooxml/e-w_tblpPr-1.html
 */
class TablePosition extends AbstractStyle
{
    /**
     * Vertical anchor constants
     *
     * @const string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_VAnchor.html
     */
    const VANCHOR_TEXT = 'text'; // Relative to vertical text extents
    const VANCHOR_MARGIN = 'margin'; // Relative to margin
    const VANCHOR_PAGE = 'page'; // Relative to page

    /**
     * Horizontal anchor constants
     *
     * @const string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_HAnchor.html
     */
    const HANCHOR_TEXT = 'text'; // Relative to text extents
    const HANCHOR_MARGIN = 'margin'; // Relative to margin
    const HANCHOR_PAGE = 'page'; // Relative to page

    /**
     * Horizontal alignment constants
     *
     * @const string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_XAlign.html
     */
    const XALIGN_LEFT = 'left'; // Left aligned horizontally
    const XALIGN_CENTER = 'center'; // Centered horizontally
    const XALIGN_RIGHT = 'right'; // Right aligned horizontally
    const XALIGN_INSIDE = 'inside'; // Inside
    const XALIGN_OUTSIDE = 'outside'; // Outside

    /**
     * Vertical alignment constants
     *
     * @const string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_YAlign.html
     */
    const YALIGN_INLINE = 'inline'; // In line with text
    const YALIGN_TOP = 'top'; // Top
    const YALIGN_CENTER = 'center'; // Centered vertically
    const YALIGN_BOTTOM = 'bottom'; // Bottom
    const YALIGN_INSIDE = 'inside'; // Inside Anchor Extents
    const YALIGN_OUTSIDE = 'outside'; // Centered vertically

    /**
     * Distance from left of table to text
     *
     * @var int
     */
    private $leftFromText;

    /**
     * Distance from right of table to text
     *
     * @var int
     */
    private $rightFromText;

    /**
     * Distance from top of table to text
     *
     * @var int
     */
    private $topFromText;

    /**
     * Distance from bottom of table to text
     *
     * @var int
     */
    private $bottomFromText;

    /**
     * Table vertical anchor
     *
     * @var string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_VAnchor.html
     */
    private $vertAnchor;

    /**
     * Table horizontal anchor
     *
     * @var string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_HAnchor.html
     */
    private $horzAnchor;

    /**
     * Relative horizontal alignment from anchor
     *
     * @var string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_XAlign.html
     */
    private $tblpXSpec;

    /**
     * Absolute horizontal distance from anchor
     *
     * @var int
     */
    private $tblpX;

    /**
     * Relative vertical alignment from anchor
     *
     * @var string
     * @see http://www.datypic.com/sc/ooxml/t-w_ST_YAlign.html
     */
    private $tblpYSpec;

    /**
     * Absolute vertical distance from anchor
     *
     * @var int
     */
    private $tblpY;

    /**
     * Create a new instance
     *
     * @param array $style
     */
    public function __construct($style = array())
    {
        $this->setStyleByArray($style);
    }

    /**
     * Get distance from left of table to text
     *
     * @return int
     */
    public function getLeftFromText()
    {
        return $this->leftFromText;
    }

    /**
     * Set distance from left of table to text
     *
     * @param int $value
     * @return self
     */
    public function setLeftFromText($value = null)
    {
        $this->leftFromText = $this->setNumericVal($value, $this->leftFromText);

        return $this;
    }

    /**
     * Get distance from right of table to text
     *
     * @return int
     */
    public function getRightFromText()
    {
        return $this->rightFromText;
    }

    /**
     * Set distance from right of table to text
     *
     * @param int $value
     * @return self
     */
    public function setRightFromText($value = null)
    {
        $this->rightFromText = $this->setNumericVal($value, $this->rightFromText);

        return $this;
    }

    /**
     * Get distance from top of table to text
     *
     * @return int
     */
    public function getTopFromText()
    {
        return $this->topFromText;
    }

    /**
     * Set distance from top of table to text
     *
     * @param int $value
     * @return self
     */
    public function setTopFromText($value = null)
    {
        $this->topFromText = $this->setNumericVal($value, $this->topFromText);

        return $this;
    }

    /**
     * Get distance from bottom of table to text
     *
     * @return int
     */
    public function getBottomFromText()
    {
        return $this->bottomFromText;
    }

    /**
     * Set distance from bottom of table to text
     *
     * @param int $value
     * @return self
     */
    public function setBottomFromText($value = null)
    {
        $this->bottomFromText = $this->setNumericVal($value, $this->bottomFromText);

        return $this;
    }

    /**
     * Get table vertical anchor
     *
     * @return string
     */
    public function getVertAnchor()
    {
        return $this->vertAnchor;
    }

    /**
     * Set table vertical anchor
     *
     * @param string $value
     * @return self
     */
    public function setVertAnchor($value = null)
    {
        $enum = array(
          self::VANCHOR_TEXT,
          self::VANCHOR_MARGIN,
          self::VANCHOR_PAGE,
        );
        $this->vertAnchor = $this->setEnumVal($value, $enum, $this->vertAnchor);

        return $this;
    }

    /**
     * Get table horizontal anchor
     *
     * @return string
     */
    public function getHorzAnchor()
    {
        return $this->horzAnchor;
    }

    /**
     * Set table horizontal anchor
     *
     * @param string $value
     * @return self
     */
    public function setHorzAnchor($value = null)
    {
        $enum = array(
          self::HANCHOR_TEXT,
          self::HANCHOR_MARGIN,
          self::HANCHOR_PAGE,
        );
        $this->horzAnchor = $this->setEnumVal($value, $enum, $this->horzAnchor);

        return $this;
    }

    /**
     * Get relative horizontal alignment from anchor
     *
     * @return string
     */
    public function getTblpXSpec()
    {
        return $this->tblpXSpec;
    }

    /**
     * Set relative horizontal alignment from anchor
     *
     * @param string $value
     * @return self
     */
    public function setTblpXSpec($value = null)
    {
        $enum = array(
            self::XALIGN_LEFT,
            self::XALIGN_CENTER,
            self::XALIGN_RIGHT,
            self::XALIGN_INSIDE,
            self::XALIGN_OUTSIDE,
        );
        $this->tblpXSpec = $this->setEnumVal($value, $enum, $this->tblpXSpec);

        return $this;
    }

    /**
     * Get absolute horizontal distance from anchor
     *
     * @return int
     */
    public function getTblpX()
    {
        return $this->tblpX;
    }

    /**
     * Set absolute horizontal distance from anchor
     *
     * @param int $value
     * @return self
     */
    public function setTblpX($value = null)
    {
        $this->tblpX = $this->setNumericVal($value, $this->tblpX);

        return $this;
    }

    /**
     * Get relative vertical alignment from anchor
     *
     * @return string
     */
    public function getTblpYSpec()
    {
        return $this->tblpYSpec;
    }

    /**
     * Set relative vertical alignment from anchor
     *
     * @param string $value
     * @return self
     */
    public function setTblpYSpec($value = null)
    {
        $enum = array(
            self::YALIGN_INLINE,
            self::YALIGN_TOP,
            self::YALIGN_CENTER,
            self::YALIGN_BOTTOM,
            self::YALIGN_INSIDE,
            self::YALIGN_OUTSIDE,
        );
        $this->tblpYSpec = $this->setEnumVal($value, $enum, $this->tblpYSpec);

        return $this;
    }

    /**
     * Get absolute vertical distance from anchor
     *
     * @return int
     */
    public function getTblpY()
    {
        return $this->tblpY;
    }

    /**
     * Set absolute vertical distance from anchor
     *
     * @param int $value
     * @return self
     */
    public function setTblpY($value = null)
    {
        $this->tblpY = $this->setNumericVal($value, $this->tblpY);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/TextBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * TextBox style
 *
 * @since 0.11.0
 */
class TextBox extends Image
{
    /**
     * margin top
     *
     * @var int
     */
    private $innerMarginTop = null;

    /**
     * margin left
     *
     * @var int
     */
    private $innerMarginLeft = null;

    /**
     * margin right
     *
     * @var int
     */
    private $innerMarginRight = null;

    /**
     * Cell margin bottom
     *
     * @var int
     */
    private $innerMarginBottom = null;

    /**
     * border size
     *
     * @var int
     */
    private $borderSize = null;

    /**
     * border color
     *
     * @var string
     */
    private $borderColor;

    /**
     * Set margin top.
     *
     * @param int $value
     */
    public function setInnerMarginTop($value = null)
    {
        $this->innerMarginTop = $value;
    }

    /**
     * Get margin top
     *
     * @return int
     */
    public function getInnerMarginTop()
    {
        return $this->innerMarginTop;
    }

    /**
     * Set margin left.
     *
     * @param int $value
     */
    public function setInnerMarginLeft($value = null)
    {
        $this->innerMarginLeft = $value;
    }

    /**
     * Get margin left
     *
     * @return int
     */
    public function getInnerMarginLeft()
    {
        return $this->innerMarginLeft;
    }

    /**
     * Set margin right.
     *
     * @param int $value
     */
    public function setInnerMarginRight($value = null)
    {
        $this->innerMarginRight = $value;
    }

    /**
     * Get margin right
     *
     * @return int
     */
    public function getInnerMarginRight()
    {
        return $this->innerMarginRight;
    }

    /**
     * Set margin bottom.
     *
     * @param int $value
     */
    public function setInnerMarginBottom($value = null)
    {
        $this->innerMarginBottom = $value;
    }

    /**
     * Get margin bottom
     *
     * @return int
     */
    public function getInnerMarginBottom()
    {
        return $this->innerMarginBottom;
    }

    /**
     * Set TLRB cell margin.
     *
     * @param int $value Margin in twips
     */
    public function setInnerMargin($value = null)
    {
        $this->setInnerMarginTop($value);
        $this->setInnerMarginLeft($value);
        $this->setInnerMarginRight($value);
        $this->setInnerMarginBottom($value);
    }

    /**
     * Get cell margin
     *
     * @return int[]
     */
    public function getInnerMargin()
    {
        return array($this->innerMarginLeft, $this->innerMarginTop, $this->innerMarginRight, $this->innerMarginBottom);
    }

    /**
     * Has inner margin?
     *
     * @return bool
     */
    public function hasInnerMargins()
    {
        $hasInnerMargins = false;
        $margins = $this->getInnerMargin();
        $numMargins = count($margins);
        for ($i = 0; $i < $numMargins; $i++) {
            if ($margins[$i] !== null) {
                $hasInnerMargins = true;
            }
        }

        return $hasInnerMargins;
    }

    /**
     * Set border size.
     *
     * @param int $value Size in points
     */
    public function setBorderSize($value = null)
    {
        $this->borderSize = $value;
    }

    /**
     * Get border size
     *
     * @return int
     */
    public function getBorderSize()
    {
        return $this->borderSize;
    }

    /**
     * Set border color.
     *
     * @param string $value
     */
    public function setBorderColor($value = null)
    {
        $this->borderColor = $value;
    }

    /**
     * Get border color
     *
     * @return string
     */
    public function getBorderColor()
    {
        return $this->borderColor;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style/TOC.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Style;

/**
 * TOC style
 */
class TOC extends Tab
{
    /**
     * Tab leader types for backward compatibility
     *
     * @deprecated 0.11.0
     *
     * @const string
     */
    const TABLEADER_DOT = self::TAB_LEADER_DOT;
    const TABLEADER_UNDERSCORE = self::TAB_LEADER_UNDERSCORE;
    const TABLEADER_LINE = self::TAB_LEADER_HYPHEN;
    const TABLEADER_NONE = self::TAB_LEADER_NONE;

    /**
     * Indent
     *
     * @var int|float (twip)
     */
    private $indent = 200;

    /**
     * Create a new TOC Style
     */
    public function __construct()
    {
        parent::__construct(self::TAB_STOP_RIGHT, 9062, self::TAB_LEADER_DOT);
    }

    /**
     * Get Tab Position
     *
     * @return int|float
     */
    public function getTabPos()
    {
        return $this->getPosition();
    }

    /**
     * Set Tab Position
     *
     * @param int|float $value
     * @return self
     */
    public function setTabPos($value)
    {
        return $this->setPosition($value);
    }

    /**
     * Get Tab Leader
     *
     * @return string
     */
    public function getTabLeader()
    {
        return $this->getLeader();
    }

    /**
     * Set Tab Leader
     *
     * @param string $value
     * @return self
     */
    public function setTabLeader($value = self::TAB_LEADER_DOT)
    {
        return $this->setLeader($value);
    }

    /**
     * Get Indent
     *
     * @return int|float
     */
    public function getIndent()
    {
        return $this->indent;
    }

    /**
     * Set Indent
     *
     * @param int|float $value
     * @return self
     */
    public function setIndent($value)
    {
        $this->indent = $this->setNumericVal($value, $this->indent);

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Style.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

use PhpOffice\PhpWord\Style\AbstractStyle;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Numbering;
use PhpOffice\PhpWord\Style\Paragraph;
use PhpOffice\PhpWord\Style\Table;

/**
 * Style collection
 */
class Style
{
    /**
     * Style register
     *
     * @var array
     */
    private static $styles = array();

    /**
     * Add paragraph style
     *
     * @param string $styleName
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $styles
     * @return \PhpOffice\PhpWord\Style\Paragraph
     */
    public static function addParagraphStyle($styleName, $styles)
    {
        return self::setStyleValues($styleName, new Paragraph(), $styles);
    }

    /**
     * Add font style
     *
     * @param string $styleName
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $fontStyle
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $paragraphStyle
     * @return \PhpOffice\PhpWord\Style\Font
     */
    public static function addFontStyle($styleName, $fontStyle, $paragraphStyle = null)
    {
        return self::setStyleValues($styleName, new Font('text', $paragraphStyle), $fontStyle);
    }

    /**
     * Add link style
     *
     * @param string $styleName
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $styles
     * @return \PhpOffice\PhpWord\Style\Font
     */
    public static function addLinkStyle($styleName, $styles)
    {
        return self::setStyleValues($styleName, new Font('link'), $styles);
    }

    /**
     * Add numbering style
     *
     * @param string $styleName
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $styleValues
     * @return \PhpOffice\PhpWord\Style\Numbering
     * @since 0.10.0
     */
    public static function addNumberingStyle($styleName, $styleValues)
    {
        return self::setStyleValues($styleName, new Numbering(), $styleValues);
    }

    /**
     * Add title style
     *
     * @param int|null $depth Provide null to set title font
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $fontStyle
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $paragraphStyle
     * @return \PhpOffice\PhpWord\Style\Font
     */
    public static function addTitleStyle($depth, $fontStyle, $paragraphStyle = null)
    {
        if (empty($depth)) {
            $styleName = 'Title';
        } else {
            $styleName = "Heading_{$depth}";
        }

        return self::setStyleValues($styleName, new Font('title', $paragraphStyle), $fontStyle);
    }

    /**
     * Add table style
     *
     * @param string $styleName
     * @param array $styleTable
     * @param array|null $styleFirstRow
     * @return \PhpOffice\PhpWord\Style\Table
     */
    public static function addTableStyle($styleName, $styleTable, $styleFirstRow = null)
    {
        return self::setStyleValues($styleName, new Table($styleTable, $styleFirstRow), null);
    }

    /**
     * Count styles
     *
     * @return int
     * @since 0.10.0
     */
    public static function countStyles()
    {
        return count(self::$styles);
    }

    /**
     * Reset styles.
     *
     * @since 0.10.0
     */
    public static function resetStyles()
    {
        self::$styles = array();
    }

    /**
     * Set default paragraph style
     *
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $styles Paragraph style definition
     * @return \PhpOffice\PhpWord\Style\Paragraph
     */
    public static function setDefaultParagraphStyle($styles)
    {
        return self::addParagraphStyle('Normal', $styles);
    }

    /**
     * Get all styles
     *
     * @return \PhpOffice\PhpWord\Style\AbstractStyle[]
     */
    public static function getStyles()
    {
        return self::$styles;
    }

    /**
     * Get style by name
     *
     * @param string $styleName
     * @return \PhpOffice\PhpWord\Style\AbstractStyle Paragraph|Font|Table|Numbering
     */
    public static function getStyle($styleName)
    {
        if (isset(self::$styles[$styleName])) {
            return self::$styles[$styleName];
        }

        return null;
    }

    /**
     * Set style values and put it to static style collection
     *
     * The $styleValues could be an array or object
     *
     * @param string $name
     * @param \PhpOffice\PhpWord\Style\AbstractStyle $style
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $value
     * @return \PhpOffice\PhpWord\Style\AbstractStyle
     */
    private static function setStyleValues($name, $style, $value = null)
    {
        if (!isset(self::$styles[$name])) {
            if ($value !== null) {
                if (is_array($value)) {
                    $style->setStyleByArray($value);
                } elseif ($value instanceof AbstractStyle) {
                    if (get_class($style) == get_class($value)) {
                        $style = $value;
                    }
                }
            }
            $style->setStyleName($name);
            $style->setIndex(self::countStyles() + 1); // One based index
            self::$styles[$name] = $style;
        }

        return self::getStyle($name);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Template.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

/**
 * @deprecated 0.12.0 Use `\PhpOffice\PhpWord\TemplateProcessor` instead.
 *
 * @codeCoverageIgnore
 */
class Template extends TemplateProcessor
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/TemplateProcessor.php

<?php

/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord;

use PhpOffice\PhpWord\Escaper\RegExp;
use PhpOffice\PhpWord\Escaper\Xml;
use PhpOffice\PhpWord\Exception\CopyFileException;
use PhpOffice\PhpWord\Exception\CreateTemporaryFileException;
use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Shared\Text;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Shared\ZipArchive;

class TemplateProcessor
{
    const MAXIMUM_REPLACEMENTS_DEFAULT = -1;

    /**
     * ZipArchive object.
     *
     * @var mixed
     */
    protected $zipClass;

    /**
     * @var string Temporary document filename (with path)
     */
    protected $tempDocumentFilename;

    /**
     * Content of main document part (in XML format) of the temporary document
     *
     * @var string
     */
    protected $tempDocumentMainPart;

    /**
     * Content of settings part (in XML format) of the temporary document
     *
     * @var string
     */
    protected $tempDocumentSettingsPart;

    /**
     * Content of headers (in XML format) of the temporary document
     *
     * @var string[]
     */
    protected $tempDocumentHeaders = array();

    /**
     * Content of footers (in XML format) of the temporary document
     *
     * @var string[]
     */
    protected $tempDocumentFooters = array();

    /**
     * Document relations (in XML format) of the temporary document.
     *
     * @var string[]
     */
    protected $tempDocumentRelations = array();

    /**
     * Document content types (in XML format) of the temporary document.
     *
     * @var string
     */
    protected $tempDocumentContentTypes = '';

    /**
     * new inserted images list
     *
     * @var string[]
     */
    protected $tempDocumentNewImages = array();

    /**
     * @since 0.12.0 Throws CreateTemporaryFileException and CopyFileException instead of Exception
     *
     * @param string $documentTemplate The fully qualified template filename
     *
     * @throws \PhpOffice\PhpWord\Exception\CreateTemporaryFileException
     * @throws \PhpOffice\PhpWord\Exception\CopyFileException
     */
    public function __construct($documentTemplate)
    {
        // Temporary document filename initialization
        $this->tempDocumentFilename = tempnam(Settings::getTempDir(), 'PhpWord');
        if (false === $this->tempDocumentFilename) {
            throw new CreateTemporaryFileException(); // @codeCoverageIgnore
        }

        // Template file cloning
        if (false === copy($documentTemplate, $this->tempDocumentFilename)) {
            throw new CopyFileException($documentTemplate, $this->tempDocumentFilename); // @codeCoverageIgnore
        }

        // Temporary document content extraction
        $this->zipClass = new ZipArchive();
        $this->zipClass->open($this->tempDocumentFilename);
        $index = 1;
        while (false !== $this->zipClass->locateName($this->getHeaderName($index))) {
            $this->tempDocumentHeaders[$index] = $this->readPartWithRels($this->getHeaderName($index));
            $index++;
        }
        $index = 1;
        while (false !== $this->zipClass->locateName($this->getFooterName($index))) {
            $this->tempDocumentFooters[$index] = $this->readPartWithRels($this->getFooterName($index));
            $index++;
        }

        $this->tempDocumentMainPart = $this->readPartWithRels($this->getMainPartName());
        $this->tempDocumentSettingsPart = $this->readPartWithRels($this->getSettingsPartName());
        $this->tempDocumentContentTypes = $this->zipClass->getFromName($this->getDocumentContentTypesName());
    }

    /**
     * Expose zip class
     *
     * To replace an image: $templateProcessor->zip()->AddFromString("word/media/image1.jpg", file_get_contents($file));<br>
     * To read a file: $templateProcessor->zip()->getFromName("word/media/image1.jpg");
     *
     * @return \PhpOffice\PhpWord\Shared\ZipArchive
     */
    public function zip()
    {
        return $this->zipClass;
    }

    /**
     * @param string $fileName
     *
     * @return string
     */
    protected function readPartWithRels($fileName)
    {
        $relsFileName = $this->getRelationsName($fileName);
        $partRelations = $this->zipClass->getFromName($relsFileName);
        if ($partRelations !== false) {
            $this->tempDocumentRelations[$fileName] = $partRelations;
        }

        return $this->fixBrokenMacros($this->zipClass->getFromName($fileName));
    }

    /**
     * @param string $xml
     * @param \XSLTProcessor $xsltProcessor
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return string
     */
    protected function transformSingleXml($xml, $xsltProcessor)
    {
        if (\PHP_VERSION_ID < 80000) {
            $orignalLibEntityLoader = libxml_disable_entity_loader(true);
        }
        $domDocument = new \DOMDocument();
        if (false === $domDocument->loadXML($xml)) {
            throw new Exception('Could not load the given XML document.');
        }

        $transformedXml = $xsltProcessor->transformToXml($domDocument);
        if (false === $transformedXml) {
            throw new Exception('Could not transform the given XML document.');
        }
        if (\PHP_VERSION_ID < 80000) {
            libxml_disable_entity_loader($orignalLibEntityLoader);
        }

        return $transformedXml;
    }

    /**
     * @param mixed $xml
     * @param \XSLTProcessor $xsltProcessor
     *
     * @return mixed
     */
    protected function transformXml($xml, $xsltProcessor)
    {
        if (is_array($xml)) {
            foreach ($xml as &$item) {
                $item = $this->transformSingleXml($item, $xsltProcessor);
            }
            unset($item);
        } else {
            $xml = $this->transformSingleXml($xml, $xsltProcessor);
        }

        return $xml;
    }

    /**
     * Applies XSL style sheet to template's parts.
     *
     * Note: since the method doesn't make any guess on logic of the provided XSL style sheet,
     * make sure that output is correctly escaped. Otherwise you may get broken document.
     *
     * @param \DOMDocument $xslDomDocument
     * @param array $xslOptions
     * @param string $xslOptionsUri
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function applyXslStyleSheet($xslDomDocument, $xslOptions = array(), $xslOptionsUri = '')
    {
        $xsltProcessor = new \XSLTProcessor();

        $xsltProcessor->importStylesheet($xslDomDocument);
        if (false === $xsltProcessor->setParameter($xslOptionsUri, $xslOptions)) {
            throw new Exception('Could not set values for the given XSL style sheet parameters.');
        }

        $this->tempDocumentHeaders = $this->transformXml($this->tempDocumentHeaders, $xsltProcessor);
        $this->tempDocumentMainPart = $this->transformXml($this->tempDocumentMainPart, $xsltProcessor);
        $this->tempDocumentFooters = $this->transformXml($this->tempDocumentFooters, $xsltProcessor);
    }

    /**
     * @param string $macro
     *
     * @return string
     */
    protected static function ensureMacroCompleted($macro)
    {
        if (substr($macro, 0, 2) !== '${' && substr($macro, -1) !== '}') {
            $macro = '${' . $macro . '}';
        }

        return $macro;
    }

    /**
     * @param string $subject
     *
     * @return string
     */
    protected static function ensureUtf8Encoded($subject)
    {
        if (!Text::isUTF8($subject)) {
            $subject = utf8_encode($subject);
        }

        return $subject;
    }

    /**
     * @param string $search
     * @param \PhpOffice\PhpWord\Element\AbstractElement $complexType
     */
    public function setComplexValue($search, \PhpOffice\PhpWord\Element\AbstractElement $complexType)
    {
        $elementName = substr(get_class($complexType), strrpos(get_class($complexType), '\\') + 1);
        $objectClass = 'PhpOffice\\PhpWord\\Writer\\Word2007\\Element\\' . $elementName;

        $xmlWriter = new XMLWriter();
        /** @var \PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement $elementWriter */
        $elementWriter = new $objectClass($xmlWriter, $complexType, true);
        $elementWriter->write();

        $where = $this->findContainingXmlBlockForMacro($search, 'w:r');

        if ($where === false) {
            return;
        }

        $block = $this->getSlice($where['start'], $where['end']);
        $textParts = $this->splitTextIntoTexts($block);
        $this->replaceXmlBlock($search, $textParts, 'w:r');

        $search = static::ensureMacroCompleted($search);
        $this->replaceXmlBlock($search, $xmlWriter->getData(), 'w:r');
    }

    /**
     * @param string $search
     * @param \PhpOffice\PhpWord\Element\AbstractElement $complexType
     */
    public function setComplexBlock($search, \PhpOffice\PhpWord\Element\AbstractElement $complexType)
    {
        $elementName = substr(get_class($complexType), strrpos(get_class($complexType), '\\') + 1);
        $objectClass = 'PhpOffice\\PhpWord\\Writer\\Word2007\\Element\\' . $elementName;

        $xmlWriter = new XMLWriter();
        /** @var \PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement $elementWriter */
        $elementWriter = new $objectClass($xmlWriter, $complexType, false);
        $elementWriter->write();

        $this->replaceXmlBlock($search, $xmlWriter->getData(), 'w:p');
    }

    /**
     * @param mixed $search
     * @param mixed $replace
     * @param int $limit
     */
    public function setValue($search, $replace, $limit = self::MAXIMUM_REPLACEMENTS_DEFAULT)
    {
        if (is_array($search)) {
            foreach ($search as &$item) {
                $item = static::ensureMacroCompleted($item);
            }
            unset($item);
        } else {
            $search = static::ensureMacroCompleted($search);
        }

        if (is_array($replace)) {
            foreach ($replace as &$item) {
                $item = static::ensureUtf8Encoded($item);
            }
            unset($item);
        } else {
            $replace = static::ensureUtf8Encoded($replace);
        }

        if (Settings::isOutputEscapingEnabled()) {
            $xmlEscaper = new Xml();
            $replace = $xmlEscaper->escape($replace);
        }

        $this->tempDocumentHeaders = $this->setValueForPart($search, $replace, $this->tempDocumentHeaders, $limit);
        $this->tempDocumentMainPart = $this->setValueForPart($search, $replace, $this->tempDocumentMainPart, $limit);
        $this->tempDocumentFooters = $this->setValueForPart($search, $replace, $this->tempDocumentFooters, $limit);
    }

    /**
     * Set values from a one-dimensional array of "variable => value"-pairs.
     *
     * @param array $values
     */
    public function setValues(array $values)
    {
        foreach ($values as $macro => $replace) {
            $this->setValue($macro, $replace);
        }
    }

    /**
     * @param string $search
     * @param \PhpOffice\PhpWord\Element\AbstractElement $complexType
     */
    public function setChart($search, \PhpOffice\PhpWord\Element\AbstractElement $chart)
    {
        $elementName = substr(get_class($chart), strrpos(get_class($chart), '\\') + 1);
        $objectClass = 'PhpOffice\\PhpWord\\Writer\\Word2007\\Element\\' . $elementName;

        // Get the next relation id
        $rId = $this->getNextRelationsIndex($this->getMainPartName());
        $chart->setRelationId($rId);

        // Define the chart filename
        $filename = "charts/chart{$rId}.xml";

        // Get the part writer
        $writerPart = new \PhpOffice\PhpWord\Writer\Word2007\Part\Chart();
        $writerPart->setElement($chart);

        // ContentTypes.xml
        $this->zipClass->addFromString("word/{$filename}", $writerPart->write());

        // add chart to content type
        $xmlRelationsType = "<Override PartName=\"/word/{$filename}\" ContentType=\"application/vnd.openxmlformats-officedocument.drawingml.chart+xml\"/>";
        $this->tempDocumentContentTypes = str_replace('</Types>', $xmlRelationsType, $this->tempDocumentContentTypes) . '</Types>';

        // Add the chart to relations
        $xmlChartRelation = "<Relationship Id=\"rId{$rId}\" Type=\"http://schemas.openxmlformats.org/officeDocument/2006/relationships/chart\" Target=\"charts/chart{$rId}.xml\"/>";
        $this->tempDocumentRelations[$this->getMainPartName()] = str_replace('</Relationships>', $xmlChartRelation, $this->tempDocumentRelations[$this->getMainPartName()]) . '</Relationships>';

        // Write the chart
        $xmlWriter = new XMLWriter();
        $elementWriter = new $objectClass($xmlWriter, $chart, true);
        $elementWriter->write();

        // Place it in the template
        $this->replaceXmlBlock($search, '<w:p>' . $xmlWriter->getData() . '</w:p>', 'w:p');
    }

    private function getImageArgs($varNameWithArgs)
    {
        $varElements = explode(':', $varNameWithArgs);
        array_shift($varElements); // first element is name of variable => remove it

        $varInlineArgs = array();
        // size format documentation: https://msdn.microsoft.com/en-us/library/documentformat.openxml.vml.shape%28v=office.14%29.aspx?f=255&MSPPError=-2147217396
        foreach ($varElements as $argIdx => $varArg) {
            if (strpos($varArg, '=')) { // arg=value
                list($argName, $argValue) = explode('=', $varArg, 2);
                $argName = strtolower($argName);
                if ($argName == 'size') {
                    list($varInlineArgs['width'], $varInlineArgs['height']) = explode('x', $argValue, 2);
                } else {
                    $varInlineArgs[strtolower($argName)] = $argValue;
                }
            } elseif (preg_match('/^([0-9]*[a-z%]{0,2}|auto)x([0-9]*[a-z%]{0,2}|auto)$/i', $varArg)) { // 60x40
                list($varInlineArgs['width'], $varInlineArgs['height']) = explode('x', $varArg, 2);
            } else { // :60:40:f
                switch ($argIdx) {
                    case 0:
                        $varInlineArgs['width'] = $varArg;
                        break;
                    case 1:
                        $varInlineArgs['height'] = $varArg;
                        break;
                    case 2:
                        $varInlineArgs['ratio'] = $varArg;
                        break;
                }
            }
        }

        return $varInlineArgs;
    }

    private function chooseImageDimension($baseValue, $inlineValue, $defaultValue)
    {
        $value = $baseValue;
        if (is_null($value) && isset($inlineValue)) {
            $value = $inlineValue;
        }
        if (!preg_match('/^([0-9]*(cm|mm|in|pt|pc|px|%|em|ex|)|auto)$/i', $value)) {
            $value = null;
        }
        if (is_null($value)) {
            $value = $defaultValue;
        }
        if (is_numeric($value)) {
            $value .= 'px';
        }

        return $value;
    }

    private function fixImageWidthHeightRatio(&$width, &$height, $actualWidth, $actualHeight)
    {
        $imageRatio = $actualWidth / $actualHeight;

        if (($width === '') && ($height === '')) { // defined size are empty
            $width = $actualWidth . 'px';
            $height = $actualHeight . 'px';
        } elseif ($width === '') { // defined width is empty
            $heightFloat = (float) $height;
            $widthFloat = $heightFloat * $imageRatio;
            $matches = array();
            preg_match("/\d([a-z%]+)$/", $height, $matches);
            $width = $widthFloat . $matches[1];
        } elseif ($height === '') { // defined height is empty
            $widthFloat = (float) $width;
            $heightFloat = $widthFloat / $imageRatio;
            $matches = array();
            preg_match("/\d([a-z%]+)$/", $width, $matches);
            $height = $heightFloat . $matches[1];
        } else { // we have defined size, but we need also check it aspect ratio
            $widthMatches = array();
            preg_match("/\d([a-z%]+)$/", $width, $widthMatches);
            $heightMatches = array();
            preg_match("/\d([a-z%]+)$/", $height, $heightMatches);
            // try to fix only if dimensions are same
            if ($widthMatches[1] == $heightMatches[1]) {
                $dimention = $widthMatches[1];
                $widthFloat = (float) $width;
                $heightFloat = (float) $height;
                $definedRatio = $widthFloat / $heightFloat;

                if ($imageRatio > $definedRatio) { // image wider than defined box
                    $height = ($widthFloat / $imageRatio) . $dimention;
                } elseif ($imageRatio < $definedRatio) { // image higher than defined box
                    $width = ($heightFloat * $imageRatio) . $dimention;
                }
            }
        }
    }

    private function prepareImageAttrs($replaceImage, $varInlineArgs)
    {
        // get image path and size
        $width = null;
        $height = null;
        $ratio = null;

        // a closure can be passed as replacement value which after resolving, can contain the replacement info for the image
        // use case: only when a image if found, the replacement tags can be generated
        if (is_callable($replaceImage)) {
            $replaceImage = $replaceImage();
        }

        if (is_array($replaceImage) && isset($replaceImage['path'])) {
            $imgPath = $replaceImage['path'];
            if (isset($replaceImage['width'])) {
                $width = $replaceImage['width'];
            }
            if (isset($replaceImage['height'])) {
                $height = $replaceImage['height'];
            }
            if (isset($replaceImage['ratio'])) {
                $ratio = $replaceImage['ratio'];
            }
        } else {
            $imgPath = $replaceImage;
        }

        $width = $this->chooseImageDimension($width, isset($varInlineArgs['width']) ? $varInlineArgs['width'] : null, 115);
        $height = $this->chooseImageDimension($height, isset($varInlineArgs['height']) ? $varInlineArgs['height'] : null, 70);

        $imageData = @getimagesize($imgPath);
        if (!is_array($imageData)) {
            throw new Exception(sprintf('Invalid image: %s', $imgPath));
        }
        list($actualWidth, $actualHeight, $imageType) = $imageData;

        // fix aspect ratio (by default)
        if (is_null($ratio) && isset($varInlineArgs['ratio'])) {
            $ratio = $varInlineArgs['ratio'];
        }
        if (is_null($ratio) || !in_array(strtolower($ratio), array('', '-', 'f', 'false'))) {
            $this->fixImageWidthHeightRatio($width, $height, $actualWidth, $actualHeight);
        }

        $imageAttrs = array(
            'src'    => $imgPath,
            'mime'   => image_type_to_mime_type($imageType),
            'width'  => $width,
            'height' => $height,
        );

        return $imageAttrs;
    }

    private function addImageToRelations($partFileName, $rid, $imgPath, $imageMimeType)
    {
        // define templates
        $typeTpl = '<Override PartName="/word/media/{IMG}" ContentType="image/{EXT}"/>';
        $relationTpl = '<Relationship Id="{RID}" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/{IMG}"/>';
        $newRelationsTpl = '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>' . "\n" . '<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"></Relationships>';
        $newRelationsTypeTpl = '<Override PartName="/{RELS}" ContentType="application/vnd.openxmlformats-package.relationships+xml"/>';
        $extTransform = array(
            'image/jpeg' => 'jpeg',
            'image/png'  => 'png',
            'image/bmp'  => 'bmp',
            'image/gif'  => 'gif',
        );

        // get image embed name
        if (isset($this->tempDocumentNewImages[$imgPath])) {
            $imgName = $this->tempDocumentNewImages[$imgPath];
        } else {
            // transform extension
            if (isset($extTransform[$imageMimeType])) {
                $imgExt = $extTransform[$imageMimeType];
            } else {
                throw new Exception("Unsupported image type $imageMimeType");
            }

            // add image to document
            $imgName = 'image_' . $rid . '_' . pathinfo($partFileName, PATHINFO_FILENAME) . '.' . $imgExt;
            $this->zipClass->pclzipAddFile($imgPath, 'word/media/' . $imgName);
            $this->tempDocumentNewImages[$imgPath] = $imgName;

            // setup type for image
            $xmlImageType = str_replace(array('{IMG}', '{EXT}'), array($imgName, $imgExt), $typeTpl);
            $this->tempDocumentContentTypes = str_replace('</Types>', $xmlImageType, $this->tempDocumentContentTypes) . '</Types>';
        }

        $xmlImageRelation = str_replace(array('{RID}', '{IMG}'), array($rid, $imgName), $relationTpl);

        if (!isset($this->tempDocumentRelations[$partFileName])) {
            // create new relations file
            $this->tempDocumentRelations[$partFileName] = $newRelationsTpl;
            // and add it to content types
            $xmlRelationsType = str_replace('{RELS}', $this->getRelationsName($partFileName), $newRelationsTypeTpl);
            $this->tempDocumentContentTypes = str_replace('</Types>', $xmlRelationsType, $this->tempDocumentContentTypes) . '</Types>';
        }

        // add image to relations
        $this->tempDocumentRelations[$partFileName] = str_replace('</Relationships>', $xmlImageRelation, $this->tempDocumentRelations[$partFileName]) . '</Relationships>';
    }

    /**
     * @param mixed $search
     * @param mixed $replace Path to image, or array("path" => xx, "width" => yy, "height" => zz)
     * @param int $limit
     */
    public function setImageValue($search, $replace, $limit = self::MAXIMUM_REPLACEMENTS_DEFAULT)
    {
        // prepare $search_replace
        if (!is_array($search)) {
            $search = array($search);
        }

        $replacesList = array();
        if (!is_array($replace) || isset($replace['path'])) {
            $replacesList[] = $replace;
        } else {
            $replacesList = array_values($replace);
        }

        $searchReplace = array();
        foreach ($search as $searchIdx => $searchString) {
            $searchReplace[$searchString] = isset($replacesList[$searchIdx]) ? $replacesList[$searchIdx] : $replacesList[0];
        }

        // collect document parts
        $searchParts = array(
            $this->getMainPartName() => &$this->tempDocumentMainPart,
        );
        foreach (array_keys($this->tempDocumentHeaders) as $headerIndex) {
            $searchParts[$this->getHeaderName($headerIndex)] = &$this->tempDocumentHeaders[$headerIndex];
        }
        foreach (array_keys($this->tempDocumentFooters) as $headerIndex) {
            $searchParts[$this->getFooterName($headerIndex)] = &$this->tempDocumentFooters[$headerIndex];
        }

        // define templates
        // result can be verified via "Open XML SDK 2.5 Productivity Tool" (http://www.microsoft.com/en-us/download/details.aspx?id=30425)
        $imgTpl = '<w:pict><v:shape type="#_x0000_t75" style="width:{WIDTH};height:{HEIGHT}" stroked="f"><v:imagedata r:id="{RID}" o:title=""/></v:shape></w:pict>';

        $i = 0;
        foreach ($searchParts as $partFileName => &$partContent) {
            $partVariables = $this->getVariablesForPart($partContent);

            foreach ($searchReplace as $searchString => $replaceImage) {
                $varsToReplace = array_filter($partVariables, function ($partVar) use ($searchString) {
                    return ($partVar == $searchString) || preg_match('/^' . preg_quote($searchString) . ':/', $partVar);
                });

                foreach ($varsToReplace as $varNameWithArgs) {
                    $varInlineArgs = $this->getImageArgs($varNameWithArgs);
                    $preparedImageAttrs = $this->prepareImageAttrs($replaceImage, $varInlineArgs);
                    $imgPath = $preparedImageAttrs['src'];

                    // get image index
                    $imgIndex = $this->getNextRelationsIndex($partFileName);
                    $rid = 'rId' . $imgIndex;

                    // replace preparations
                    $this->addImageToRelations($partFileName, $rid, $imgPath, $preparedImageAttrs['mime']);
                    $xmlImage = str_replace(array('{RID}', '{WIDTH}', '{HEIGHT}'), array($rid, $preparedImageAttrs['width'], $preparedImageAttrs['height']), $imgTpl);

                    // replace variable
                    $varNameWithArgsFixed = static::ensureMacroCompleted($varNameWithArgs);
                    $matches = array();
                    if (preg_match('/(<[^<]+>)([^<]*)(' . preg_quote($varNameWithArgsFixed) . ')([^>]*)(<[^>]+>)/Uu', $partContent, $matches)) {
                        $wholeTag = $matches[0];
                        array_shift($matches);
                        list($openTag, $prefix, , $postfix, $closeTag) = $matches;
                        $replaceXml = $openTag . $prefix . $closeTag . $xmlImage . $openTag . $postfix . $closeTag;
                        // replace on each iteration, because in one tag we can have 2+ inline variables => before proceed next variable we need to change $partContent
                        $partContent = $this->setValueForPart($wholeTag, $replaceXml, $partContent, $limit);
                    }

                    if (++$i >= $limit) {
                        break;
                    }
                }
            }
        }
    }

    /**
     * Returns count of all variables in template.
     *
     * @return array
     */
    public function getVariableCount()
    {
        $variables = $this->getVariablesForPart($this->tempDocumentMainPart);

        foreach ($this->tempDocumentHeaders as $headerXML) {
            $variables = array_merge(
                $variables,
                $this->getVariablesForPart($headerXML)
            );
        }

        foreach ($this->tempDocumentFooters as $footerXML) {
            $variables = array_merge(
                $variables,
                $this->getVariablesForPart($footerXML)
            );
        }

        return array_count_values($variables);
    }

    /**
     * Returns array of all variables in template.
     *
     * @return string[]
     */
    public function getVariables()
    {
        return array_keys($this->getVariableCount());
    }

    /**
     * Clone a table row in a template document.
     *
     * @param string $search
     * @param int $numberOfClones
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function cloneRow($search, $numberOfClones)
    {
        $search = static::ensureMacroCompleted($search);

        $tagPos = strpos($this->tempDocumentMainPart, $search);
        if (!$tagPos) {
            throw new Exception('Can not clone row, template variable not found or variable contains markup.');
        }

        $rowStart = $this->findRowStart($tagPos);
        $rowEnd = $this->findRowEnd($tagPos);
        $xmlRow = $this->getSlice($rowStart, $rowEnd);

        // Check if there's a cell spanning multiple rows.
        if (preg_match('#<w:vMerge w:val="restart"/>#', $xmlRow)) {
            // $extraRowStart = $rowEnd;
            $extraRowEnd = $rowEnd;
            while (true) {
                $extraRowStart = $this->findRowStart($extraRowEnd + 1);
                $extraRowEnd = $this->findRowEnd($extraRowEnd + 1);

                // If extraRowEnd is lower then 7, there was no next row found.
                if ($extraRowEnd < 7) {
                    break;
                }

                // If tmpXmlRow doesn't contain continue, this row is no longer part of the spanned row.
                $tmpXmlRow = $this->getSlice($extraRowStart, $extraRowEnd);
                if (!preg_match('#<w:vMerge/>#', $tmpXmlRow) &&
                    !preg_match('#<w:vMerge w:val="continue"\s*/>#', $tmpXmlRow)
                ) {
                    break;
                }
                // This row was a spanned row, update $rowEnd and search for the next row.
                $rowEnd = $extraRowEnd;
            }
            $xmlRow = $this->getSlice($rowStart, $rowEnd);
        }

        $result = $this->getSlice(0, $rowStart);
        $result .= implode($this->indexClonedVariables($numberOfClones, $xmlRow));
        $result .= $this->getSlice($rowEnd);

        $this->tempDocumentMainPart = $result;
    }

    /**
     * Clones a table row and populates it's values from a two-dimensional array in a template document.
     *
     * @param string $search
     * @param array $values
     */
    public function cloneRowAndSetValues($search, $values)
    {
        $this->cloneRow($search, count($values));

        foreach ($values as $rowKey => $rowData) {
            $rowNumber = $rowKey + 1;
            foreach ($rowData as $macro => $replace) {
                $this->setValue($macro . '#' . $rowNumber, $replace);
            }
        }
    }

    /**
     * Clone a block.
     *
     * @param string $blockname
     * @param int $clones How many time the block should be cloned
     * @param bool $replace
     * @param bool $indexVariables If true, any variables inside the block will be indexed (postfixed with #1, #2, ...)
     * @param array $variableReplacements Array containing replacements for macros found inside the block to clone
     *
     * @return string|null
     */
    public function cloneBlock($blockname, $clones = 1, $replace = true, $indexVariables = false, $variableReplacements = null)
    {
        $xmlBlock = null;
        $matches = array();
        preg_match(
            '/(.*((?s)<w:p\b(?:(?!<w:p\b).)*?\${' . $blockname . '}<\/w:.*?p>))(.*)((?s)<w:p\b(?:(?!<w:p\b).)[^$]*?\${\/' . $blockname . '}<\/w:.*?p>)/is',
            $this->tempDocumentMainPart,
            $matches
        );

        if (isset($matches[3])) {
            $xmlBlock = $matches[3];
            if ($indexVariables) {
                $cloned = $this->indexClonedVariables($clones, $xmlBlock);
            } elseif ($variableReplacements !== null && is_array($variableReplacements)) {
                $cloned = $this->replaceClonedVariables($variableReplacements, $xmlBlock);
            } else {
                $cloned = array();
                for ($i = 1; $i <= $clones; $i++) {
                    $cloned[] = $xmlBlock;
                }
            }

            if ($replace) {
                $this->tempDocumentMainPart = str_replace(
                    $matches[2] . $matches[3] . $matches[4],
                    implode('', $cloned),
                    $this->tempDocumentMainPart
                );
            }
        }

        return $xmlBlock;
    }

    /**
     * Replace a block.
     *
     * @param string $blockname
     * @param string $replacement
     */
    public function replaceBlock($blockname, $replacement)
    {
        $matches = array();
        preg_match(
            '/(<\?xml.*)(<w:p.*>\${' . $blockname . '}<\/w:.*?p>)(.*)(<w:p.*\${\/' . $blockname . '}<\/w:.*?p>)/is',
            $this->tempDocumentMainPart,
            $matches
        );

        if (isset($matches[3])) {
            $this->tempDocumentMainPart = str_replace(
                $matches[2] . $matches[3] . $matches[4],
                $replacement,
                $this->tempDocumentMainPart
            );
        }
    }

    /**
     * Delete a block of text.
     *
     * @param string $blockname
     */
    public function deleteBlock($blockname)
    {
        $this->replaceBlock($blockname, '');
    }

    /**
     * Automatically Recalculate Fields on Open
     *
     * @param bool $update
     */
    public function setUpdateFields($update = true)
    {
        $string = $update ? 'true' : 'false';
        $matches = array();
        if (preg_match('/<w:updateFields w:val=\"(true|false|1|0|on|off)\"\/>/', $this->tempDocumentSettingsPart, $matches)) {
            $this->tempDocumentSettingsPart = str_replace($matches[0], '<w:updateFields w:val="' . $string . '"/>', $this->tempDocumentSettingsPart);
        } else {
            $this->tempDocumentSettingsPart = str_replace('</w:settings>', '<w:updateFields w:val="' . $string . '"/></w:settings>', $this->tempDocumentSettingsPart);
        }
    }

    /**
     * Saves the result document.
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return string
     */
    public function save()
    {
        foreach ($this->tempDocumentHeaders as $index => $xml) {
            $this->savePartWithRels($this->getHeaderName($index), $xml);
        }

        $this->savePartWithRels($this->getMainPartName(), $this->tempDocumentMainPart);
        $this->savePartWithRels($this->getSettingsPartName(), $this->tempDocumentSettingsPart);

        foreach ($this->tempDocumentFooters as $index => $xml) {
            $this->savePartWithRels($this->getFooterName($index), $xml);
        }

        $this->zipClass->addFromString($this->getDocumentContentTypesName(), $this->tempDocumentContentTypes);

        // Close zip file
        if (false === $this->zipClass->close()) {
            throw new Exception('Could not close zip file.'); // @codeCoverageIgnore
        }

        return $this->tempDocumentFilename;
    }

    /**
     * @param string $fileName
     * @param string $xml
     */
    protected function savePartWithRels($fileName, $xml)
    {
        $this->zipClass->addFromString($fileName, $xml);
        if (isset($this->tempDocumentRelations[$fileName])) {
            $relsFileName = $this->getRelationsName($fileName);
            $this->zipClass->addFromString($relsFileName, $this->tempDocumentRelations[$fileName]);
        }
    }

    /**
     * Saves the result document to the user defined file.
     *
     * @since 0.8.0
     *
     * @param string $fileName
     */
    public function saveAs($fileName)
    {
        $tempFileName = $this->save();

        if (file_exists($fileName)) {
            unlink($fileName);
        }

        /*
         * Note: we do not use `rename` function here, because it loses file ownership data on Windows platform.
         * As a result, user cannot open the file directly getting "Access denied" message.
         *
         * @see https://github.com/PHPOffice/PHPWord/issues/532
         */
        copy($tempFileName, $fileName);
        unlink($tempFileName);
    }

    /**
     * Finds parts of broken macros and sticks them together.
     * Macros, while being edited, could be implicitly broken by some of the word processors.
     *
     * @param string $documentPart The document part in XML representation
     *
     * @return string
     */
    protected function fixBrokenMacros($documentPart)
    {
        return preg_replace_callback(
            '/\$(?:\{|[^{$]*\>\{)[^}$]*\}/U',
            function ($match) {
                return strip_tags($match[0]);
            },
            $documentPart
        );
    }

    /**
     * Find and replace macros in the given XML section.
     *
     * @param mixed $search
     * @param mixed $replace
     * @param string $documentPartXML
     * @param int $limit
     *
     * @return string
     */
    protected function setValueForPart($search, $replace, $documentPartXML, $limit)
    {
        // Note: we can't use the same function for both cases here, because of performance considerations.
        if (self::MAXIMUM_REPLACEMENTS_DEFAULT === $limit) {
            return str_replace($search, $replace, $documentPartXML);
        }
        $regExpEscaper = new RegExp();

        return preg_replace($regExpEscaper->escape($search), $replace, $documentPartXML, $limit);
    }

    /**
     * Find all variables in $documentPartXML.
     *
     * @param string $documentPartXML
     *
     * @return string[]
     */
    protected function getVariablesForPart($documentPartXML)
    {
        $matches = array();
        preg_match_all('/\$\{(.*?)}/i', $documentPartXML, $matches);

        return $matches[1];
    }

    /**
     * Get the name of the header file for $index.
     *
     * @param int $index
     *
     * @return string
     */
    protected function getHeaderName($index)
    {
        return sprintf('word/header%d.xml', $index);
    }

    /**
     * Usually, the name of main part document will be 'document.xml'. However, some .docx files (possibly those from Office 365, experienced also on documents from Word Online created from blank templates) have file 'document22.xml' in their zip archive instead of 'document.xml'. This method searches content types file to correctly determine the file name.
     *
     * @return string
     */
    protected function getMainPartName()
    {
        $contentTypes = $this->zipClass->getFromName('[Content_Types].xml');

        $pattern = '~PartName="\/(word\/document.*?\.xml)" ContentType="application\/vnd\.openxmlformats-officedocument\.wordprocessingml\.document\.main\+xml"~';

        $matches = array();
        preg_match($pattern, $contentTypes, $matches);

        return array_key_exists(1, $matches) ? $matches[1] : 'word/document.xml';
    }

    /**
     * The name of the file containing the Settings part
     *
     * @return string
     */
    protected function getSettingsPartName()
    {
        return 'word/settings.xml';
    }

    /**
     * Get the name of the footer file for $index.
     *
     * @param int $index
     *
     * @return string
     */
    protected function getFooterName($index)
    {
        return sprintf('word/footer%d.xml', $index);
    }

    /**
     * Get the name of the relations file for document part.
     *
     * @param string $documentPartName
     *
     * @return string
     */
    protected function getRelationsName($documentPartName)
    {
        return 'word/_rels/' . pathinfo($documentPartName, PATHINFO_BASENAME) . '.rels';
    }

    protected function getNextRelationsIndex($documentPartName)
    {
        if (isset($this->tempDocumentRelations[$documentPartName])) {
            $candidate = substr_count($this->tempDocumentRelations[$documentPartName], '<Relationship');
            while (strpos($this->tempDocumentRelations[$documentPartName], 'Id="rId' . $candidate . '"') !== false) {
                $candidate++;
            }

            return $candidate;
        }

        return 1;
    }

    /**
     * @return string
     */
    protected function getDocumentContentTypesName()
    {
        return '[Content_Types].xml';
    }

    /**
     * Find the start position of the nearest table row before $offset.
     *
     * @param int $offset
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return int
     */
    protected function findRowStart($offset)
    {
        $rowStart = strrpos($this->tempDocumentMainPart, '<w:tr ', ((strlen($this->tempDocumentMainPart) - $offset) * -1));

        if (!$rowStart) {
            $rowStart = strrpos($this->tempDocumentMainPart, '<w:tr>', ((strlen($this->tempDocumentMainPart) - $offset) * -1));
        }
        if (!$rowStart) {
            throw new Exception('Can not find the start position of the row to clone.');
        }

        return $rowStart;
    }

    /**
     * Find the end position of the nearest table row after $offset.
     *
     * @param int $offset
     *
     * @return int
     */
    protected function findRowEnd($offset)
    {
        return strpos($this->tempDocumentMainPart, '</w:tr>', $offset) + 7;
    }

    /**
     * Get a slice of a string.
     *
     * @param int $startPosition
     * @param int $endPosition
     *
     * @return string
     */
    protected function getSlice($startPosition, $endPosition = 0)
    {
        if (!$endPosition) {
            $endPosition = strlen($this->tempDocumentMainPart);
        }

        return substr($this->tempDocumentMainPart, $startPosition, ($endPosition - $startPosition));
    }

    /**
     * Replaces variable names in cloned
     * rows/blocks with indexed names
     *
     * @param int $count
     * @param string $xmlBlock
     *
     * @return string
     */
    protected function indexClonedVariables($count, $xmlBlock)
    {
        $results = array();
        for ($i = 1; $i <= $count; $i++) {
            $results[] = preg_replace('/\$\{([^:]*?)(:.*?)?\}/', '\${\1#' . $i . '\2}', $xmlBlock);
        }

        return $results;
    }

    /**
     * Raplaces variables with values from array, array keys are the variable names
     *
     * @param array $variableReplacements
     * @param string $xmlBlock
     *
     * @return string[]
     */
    protected function replaceClonedVariables($variableReplacements, $xmlBlock)
    {
        $results = array();
        foreach ($variableReplacements as $replacementArray) {
            $localXmlBlock = $xmlBlock;
            foreach ($replacementArray as $search => $replacement) {
                $localXmlBlock = $this->setValueForPart(self::ensureMacroCompleted($search), $replacement, $localXmlBlock, self::MAXIMUM_REPLACEMENTS_DEFAULT);
            }
            $results[] = $localXmlBlock;
        }

        return $results;
    }

    /**
     * Replace an XML block surrounding a macro with a new block
     *
     * @param string $macro Name of macro
     * @param string $block New block content
     * @param string $blockType XML tag type of block
     * @return \PhpOffice\PhpWord\TemplateProcessor Fluent interface
     */
    public function replaceXmlBlock($macro, $block, $blockType = 'w:p')
    {
        $where = $this->findContainingXmlBlockForMacro($macro, $blockType);
        if (is_array($where)) {
            $this->tempDocumentMainPart = $this->getSlice(0, $where['start']) . $block . $this->getSlice($where['end']);
        }

        return $this;
    }

    /**
     * Find start and end of XML block containing the given macro
     * e.g. <w:p>...${macro}...</w:p>
     *
     * Note that only the first instance of the macro will be found
     *
     * @param string $macro Name of macro
     * @param string $blockType XML tag for block
     * @return bool|int[] FALSE if not found, otherwise array with start and end
     */
    protected function findContainingXmlBlockForMacro($macro, $blockType = 'w:p')
    {
        $macroPos = $this->findMacro($macro);
        if (0 > $macroPos) {
            return false;
        }
        $start = $this->findXmlBlockStart($macroPos, $blockType);
        if (0 > $start) {
            return false;
        }
        $end = $this->findXmlBlockEnd($start, $blockType);
        //if not found or if resulting string does not contain the macro we are searching for
        if (0 > $end || strstr($this->getSlice($start, $end), $macro) === false) {
            return false;
        }

        return array('start' => $start, 'end' => $end);
    }

    /**
     * Find the position of (the start of) a macro
     *
     * Returns -1 if not found, otherwise position of opening $
     *
     * Note that only the first instance of the macro will be found
     *
     * @param string $search Macro name
     * @param int $offset Offset from which to start searching
     * @return int -1 if macro not found
     */
    protected function findMacro($search, $offset = 0)
    {
        $search = static::ensureMacroCompleted($search);
        $pos = strpos($this->tempDocumentMainPart, $search, $offset);

        return ($pos === false) ? -1 : $pos;
    }

    /**
     * Find the start position of the nearest XML block start before $offset
     *
     * @param int $offset    Search position
     * @param string  $blockType XML Block tag
     * @return int -1 if block start not found
     */
    protected function findXmlBlockStart($offset, $blockType)
    {
        $reverseOffset = (strlen($this->tempDocumentMainPart) - $offset) * -1;
        // first try XML tag with attributes
        $blockStart = strrpos($this->tempDocumentMainPart, '<' . $blockType . ' ', $reverseOffset);
        // if not found, or if found but contains the XML tag without attribute
        if (false === $blockStart || strrpos($this->getSlice($blockStart, $offset), '<' . $blockType . '>')) {
            // also try XML tag without attributes
            $blockStart = strrpos($this->tempDocumentMainPart, '<' . $blockType . '>', $reverseOffset);
        }

        return ($blockStart === false) ? -1 : $blockStart;
    }

    /**
     * Find the nearest block end position after $offset
     *
     * @param int $offset    Search position
     * @param string  $blockType XML Block tag
     * @return int -1 if block end not found
     */
    protected function findXmlBlockEnd($offset, $blockType)
    {
        $blockEndStart = strpos($this->tempDocumentMainPart, '</' . $blockType . '>', $offset);
        // return position of end of tag if found, otherwise -1

        return ($blockEndStart === false) ? -1 : $blockEndStart + 3 + strlen($blockType);
    }

    /**
     * Splits a w:r/w:t into a list of w:r where each ${macro} is in a separate w:r
     *
     * @param string $text
     * @return string
     */
    protected function splitTextIntoTexts($text)
    {
        if (!$this->textNeedsSplitting($text)) {
            return $text;
        }
        $matches = array();
        if (preg_match('/(<w:rPr.*<\/w:rPr>)/i', $text, $matches)) {
            $extractedStyle = $matches[0];
        } else {
            $extractedStyle = '';
        }

        $unformattedText = preg_replace('/>\s+</', '><', $text);
        $result = str_replace(array('${', '}'), array('</w:t></w:r><w:r>' . $extractedStyle . '<w:t xml:space="preserve">${', '}</w:t></w:r><w:r>' . $extractedStyle . '<w:t xml:space="preserve">'), $unformattedText);

        return str_replace(array('<w:r>' . $extractedStyle . '<w:t xml:space="preserve"></w:t></w:r>', '<w:r><w:t xml:space="preserve"></w:t></w:r>', '<w:t>'), array('', '', '<w:t xml:space="preserve">'), $result);
    }

    /**
     * Returns true if string contains a macro that is not in it's own w:r
     *
     * @param string $text
     * @return bool
     */
    protected function textNeedsSplitting($text)
    {
        return preg_match('/[^>]\${|}[^<]/i', $text) == 1;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/AbstractWriter.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\Exception\CopyFileException;
use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\ZipArchive;

/**
 * Abstract writer class
 *
 * @since 0.10.0
 */
abstract class AbstractWriter implements WriterInterface
{
    /**
     * PHPWord object
     *
     * @var \PhpOffice\PhpWord\PhpWord
     */
    protected $phpWord = null;

    /**
     * Part name and file name pairs
     *
     * @var array
     */
    protected $parts = array();

    /**
     * Individual writers
     *
     * @var array
     */
    protected $writerParts = array();

    /**
     * Paths to store media files
     *
     * @var array
     */
    protected $mediaPaths = array('image' => '', 'object' => '');

    /**
     * Use disk caching
     *
     * @var bool
     */
    private $useDiskCaching = false;

    /**
     * Disk caching directory
     *
     * @var string
     */
    private $diskCachingDirectory = './';

    /**
     * Temporary directory
     *
     * @var string
     */
    private $tempDir = '';

    /**
     * Original file name
     *
     * @var string
     */
    private $originalFilename;

    /**
     * Temporary file name
     *
     * @var string
     */
    private $tempFilename;

    /**
     * Get PhpWord object
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     * @return \PhpOffice\PhpWord\PhpWord
     */
    public function getPhpWord()
    {
        if (!is_null($this->phpWord)) {
            return $this->phpWord;
        }
        throw new Exception('No PhpWord assigned.');
    }

    /**
     * Set PhpWord object
     *
     * @param \PhpOffice\PhpWord\PhpWord
     * @return self
     */
    public function setPhpWord(PhpWord $phpWord = null)
    {
        $this->phpWord = $phpWord;

        return $this;
    }

    /**
     * Get writer part
     *
     * @param string $partName Writer part name
     * @return mixed
     */
    public function getWriterPart($partName = '')
    {
        if ($partName != '' && isset($this->writerParts[strtolower($partName)])) {
            return $this->writerParts[strtolower($partName)];
        }

        return null;
    }

    /**
     * Get use disk caching status
     *
     * @return bool
     */
    public function isUseDiskCaching()
    {
        return $this->useDiskCaching;
    }

    /**
     * Set use disk caching status
     *
     * @param bool $value
     * @param string $directory
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     * @return self
     */
    public function setUseDiskCaching($value = false, $directory = null)
    {
        $this->useDiskCaching = $value;

        if (!is_null($directory)) {
            if (is_dir($directory)) {
                $this->diskCachingDirectory = $directory;
            } else {
                throw new Exception("Directory does not exist: $directory");
            }
        }

        return $this;
    }

    /**
     * Get disk caching directory
     *
     * @return string
     */
    public function getDiskCachingDirectory()
    {
        return $this->diskCachingDirectory;
    }

    /**
     * Get temporary directory
     *
     * @return string
     */
    public function getTempDir()
    {
        return $this->tempDir;
    }

    /**
     * Set temporary directory
     *
     * @param string $value
     * @return self
     */
    public function setTempDir($value)
    {
        if (!is_dir($value)) {
            mkdir($value);
        }
        $this->tempDir = $value;

        return $this;
    }

    /**
     * Get temporary file name
     *
     * If $filename is php://output or php://stdout, make it a temporary file
     *
     * @param string $filename
     * @return string
     */
    protected function getTempFile($filename)
    {
        // Temporary directory
        $this->setTempDir(Settings::getTempDir() . uniqid('/PHPWordWriter_', true) . '/');

        // Temporary file
        $this->originalFilename = $filename;
        if (strpos(strtolower($filename), 'php://') === 0) {
            $filename = tempnam(Settings::getTempDir(), 'PhpWord');
            if (false === $filename) {
                $filename = $this->originalFilename; // @codeCoverageIgnore
            } // @codeCoverageIgnore
        }
        $this->tempFilename = $filename;

        return $this->tempFilename;
    }

    /**
     * Cleanup temporary file.
     *
     * @throws \PhpOffice\PhpWord\Exception\CopyFileException
     */
    protected function cleanupTempFile()
    {
        if ($this->originalFilename != $this->tempFilename) {
            // @codeCoverageIgnoreStart
            // Can't find any test case. Uncomment when found.
            if (false === copy($this->tempFilename, $this->originalFilename)) {
                throw new CopyFileException($this->tempFilename, $this->originalFilename);
            }
            // @codeCoverageIgnoreEnd
            @unlink($this->tempFilename);
        }

        $this->clearTempDir();
    }

    /**
     * Clear temporary directory.
     */
    protected function clearTempDir()
    {
        if (is_dir($this->tempDir)) {
            $this->deleteDir($this->tempDir);
        }
    }

    /**
     * Get ZipArchive object
     *
     * @param string $filename
     *
     * @throws \Exception
     *
     * @return \PhpOffice\PhpWord\Shared\ZipArchive
     */
    protected function getZipArchive($filename)
    {
        // Remove any existing file
        if (file_exists($filename)) {
            unlink($filename);
        }

        // Try opening the ZIP file
        $zip = new ZipArchive();

        // @codeCoverageIgnoreStart
        // Can't find any test case. Uncomment when found.
        if ($zip->open($filename, ZipArchive::OVERWRITE) !== true) {
            if ($zip->open($filename, ZipArchive::CREATE) !== true) {
                throw new \Exception("Could not open '{$filename}' for writing.");
            }
        }
        // @codeCoverageIgnoreEnd

        return $zip;
    }

    /**
     * Open file for writing
     *
     * @since 0.11.0
     *
     * @param string $filename
     *
     * @throws \Exception
     *
     * @return resource
     */
    protected function openFile($filename)
    {
        $filename = $this->getTempFile($filename);
        $fileHandle = fopen($filename, 'w');
        // @codeCoverageIgnoreStart
        // Can't find any test case. Uncomment when found.
        if ($fileHandle === false) {
            throw new \Exception("Could not open '{$filename}' for writing.");
        }
        // @codeCoverageIgnoreEnd

        return $fileHandle;
    }

    /**
     * Write content to file.
     *
     * @since 0.11.0
     *
     * @param resource $fileHandle
     * @param string $content
     */
    protected function writeFile($fileHandle, $content)
    {
        fwrite($fileHandle, $content);
        fclose($fileHandle);
        $this->cleanupTempFile();
    }

    /**
     * Add files to package.
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param mixed $elements
     */
    protected function addFilesToPackage(ZipArchive $zip, $elements)
    {
        foreach ($elements as $element) {
            $type = $element['type']; // image|object|link

            // Skip nonregistered types and set target
            if (!isset($this->mediaPaths[$type])) {
                continue;
            }
            $target = $this->mediaPaths[$type] . $element['target'];

            // Retrive GD image content or get local media
            if (isset($element['isMemImage']) && $element['isMemImage']) {
                $image = call_user_func($element['createFunction'], $element['source']);
                if ($element['imageType'] === 'image/png') {
                    // PNG images need to preserve alpha channel information
                    imagesavealpha($image, true);
                }
                ob_start();
                call_user_func($element['imageFunction'], $image);
                $imageContents = ob_get_contents();
                ob_end_clean();
                $zip->addFromString($target, $imageContents);
                imagedestroy($image);
            } else {
                $this->addFileToPackage($zip, $element['source'], $target);
            }
        }
    }

    /**
     * Add file to package.
     *
     * Get the actual source from an archive image.
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zipPackage
     * @param string $source
     * @param string $target
     */
    protected function addFileToPackage($zipPackage, $source, $target)
    {
        $isArchive = strpos($source, 'zip://') !== false;
        $actualSource = null;
        if ($isArchive) {
            $source = substr($source, 6);
            list($zipFilename, $imageFilename) = explode('#', $source);

            $zip = new ZipArchive();
            if ($zip->open($zipFilename) !== false) {
                if ($zip->locateName($imageFilename)) {
                    $zip->extractTo($this->getTempDir(), $imageFilename);
                    $actualSource = $this->getTempDir() . DIRECTORY_SEPARATOR . $imageFilename;
                }
            }
            $zip->close();
        } else {
            $actualSource = $source;
        }

        if (!is_null($actualSource)) {
            $zipPackage->addFile($actualSource, $target);
        }
    }

    /**
     * Delete directory.
     *
     * @param string $dir
     */
    private function deleteDir($dir)
    {
        foreach (scandir($dir) as $file) {
            if ($file === '.' || $file === '..') {
                continue;
            } elseif (is_file($dir . '/' . $file)) {
                unlink($dir . '/' . $file);
            } elseif (is_dir($dir . '/' . $file)) {
                $this->deleteDir($dir . '/' . $file);
            }
        }

        rmdir($dir);
    }

    /**
     * Get use disk caching status
     *
     * @deprecated 0.10.0
     *
     * @codeCoverageIgnore
     */
    public function getUseDiskCaching()
    {
        return $this->isUseDiskCaching();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/AbstractElement.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use Laminas\Escaper\Escaper;
use PhpOffice\PhpWord\Element\AbstractElement as Element;
use PhpOffice\PhpWord\Writer\AbstractWriter;

/**
 * Abstract HTML element writer
 *
 * @since 0.11.0
 */
abstract class AbstractElement
{
    /**
     * Parent writer
     *
     * @var \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    protected $parentWriter;

    /**
     * Element
     *
     * @var \PhpOffice\PhpWord\Element\AbstractElement
     */
    protected $element;

    /**
     * Without paragraph
     *
     * @var bool
     */
    protected $withoutP = false;

    /**
     * @var \Laminas\Escaper\Escaper|\PhpOffice\PhpWord\Escaper\AbstractEscaper
     */
    protected $escaper;

    /**
     * Write element
     */
    abstract public function write();

    /**
     * Create new instance
     *
     * @param \PhpOffice\PhpWord\Writer\AbstractWriter $parentWriter
     * @param \PhpOffice\PhpWord\Element\AbstractElement $element
     * @param bool $withoutP
     */
    public function __construct(AbstractWriter $parentWriter, Element $element, $withoutP = false)
    {
        $this->parentWriter = $parentWriter;
        $this->element = $element;
        $this->withoutP = $withoutP;
        $this->escaper = new Escaper();
    }

    /**
     * Set without paragraph.
     *
     * @param bool $value
     */
    public function setWithoutP($value)
    {
        $this->withoutP = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Bookmark.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * Bookmark element HTML writer
 *
 * @since 0.15.0
 */
class Bookmark extends Text
{
    /**
     * Write bookmark
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Bookmark) {
            return '';
        }

        $content = '';
        $content .= $this->writeOpening();
        $content .= "<a name=\"{$this->element->getName()}\"/>";
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Container.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Element\AbstractContainer as ContainerElement;

/**
 * Container element HTML writer
 *
 * @since 0.11.0
 */
class Container extends AbstractElement
{
    /**
     * Namespace; Can't use __NAMESPACE__ in inherited class (RTF)
     *
     * @var string
     */
    protected $namespace = 'PhpOffice\\PhpWord\\Writer\\HTML\\Element';

    /**
     * Write container
     *
     * @return string
     */
    public function write()
    {
        $container = $this->element;
        if (!$container instanceof ContainerElement) {
            return '';
        }
        $containerClass = substr(get_class($container), strrpos(get_class($container), '\\') + 1);
        $withoutP = in_array($containerClass, array('TextRun', 'Footnote', 'Endnote')) ? true : false;
        $content = '';

        $elements = $container->getElements();
        foreach ($elements as $element) {
            $elementClass = get_class($element);
            $writerClass = str_replace('PhpOffice\\PhpWord\\Element', $this->namespace, $elementClass);
            if (class_exists($writerClass)) {
                /** @var \PhpOffice\PhpWord\Writer\HTML\Element\AbstractElement $writer Type hint */
                $writer = new $writerClass($this->parentWriter, $element, $withoutP);
                $content .= $writer->write();
            }
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Endnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * Endnote element HTML writer
 *
 * @since 0.10.0
 */
class Endnote extends Footnote
{
    /**
     * Note type
     *
     * @var string
     */
    protected $noteType = 'endnote';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Footnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * Footnote element HTML writer
 *
 * @since 0.10.0
 */
class Footnote extends AbstractElement
{
    /**
     * Note type footnote|endnote
     *
     * @var string
     */
    protected $noteType = 'footnote';

    /**
     * Write footnote/endnote marks; The actual content is written in parent writer (HTML)
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Footnote) {
            return '';
        }
        /** @var \PhpOffice\PhpWord\Writer\HTML $parentWriter Type hint */
        $parentWriter = $this->parentWriter;

        $noteId = count($parentWriter->getNotes()) + 1;
        $noteMark = $this->noteType . '-' . $this->element->getRelationId();
        $content = "<a name=\"{$noteMark}\"><a href=\"#note-{$noteId}\" class=\"NoteRef\"><sup>{$noteId}</sup></a>";

        $parentWriter->addNote($noteId, $noteMark);

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Element\Image as ImageElement;
use PhpOffice\PhpWord\Writer\HTML\Style\Image as ImageStyleWriter;

/**
 * Image element HTML writer
 *
 * @since 0.10.0
 */
class Image extends Text
{
    /**
     * Write image
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof ImageElement) {
            return '';
        }
        $content = '';
        $imageData = $this->element->getImageStringData(true);
        if ($imageData !== null) {
            $styleWriter = new ImageStyleWriter($this->element->getStyle());
            $style = $styleWriter->write();
            $imageData = 'data:' . $this->element->getImageType() . ';base64,' . $imageData;

            $content .= $this->writeOpening();
            $content .= "<img border=\"0\" style=\"{$style}\" src=\"{$imageData}\"/>";
            $content .= $this->writeClosing();
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Link.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Settings;

/**
 * Link element HTML writer
 *
 * @since 0.10.0
 */
class Link extends Text
{
    /**
     * Write link
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Link) {
            return '';
        }

        $prefix = $this->element->isInternal() ? '#' : '';
        $content = $this->writeOpening();
        if (Settings::isOutputEscapingEnabled()) {
            $content .= "<a href=\"{$prefix}{$this->escaper->escapeHtmlAttr($this->element->getSource())}\">{$this->escaper->escapeHtml($this->element->getText())}</a>";
        } else {
            $content .= "<a href=\"{$prefix}{$this->element->getSource()}\">{$this->element->getText()}</a>";
        }
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/ListItem.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Settings;

/**
 * ListItem element HTML writer
 *
 * @since 0.10.0
 */
class ListItem extends AbstractElement
{
    /**
     * Write list item
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\ListItem) {
            return '';
        }

        if (Settings::isOutputEscapingEnabled()) {
            $content = '<p>' . $this->escaper->escapeHtml($this->element->getTextObject()->getText()) . '</p>' . PHP_EOL;
        } else {
            $content = '<p>' . $this->element->getTextObject()->getText() . '</p>' . PHP_EOL;
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/ListItemRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * ListItem element HTML writer
 *
 * @since 0.10.0
 */
class ListItemRun extends TextRun
{
    /**
     * Write list item
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\ListItemRun) {
            return '';
        }

        $writer = new Container($this->parentWriter, $this->element);
        $content = $writer->write() . PHP_EOL;

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/PageBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * PageBreak element HTML writer
 *
 * @since 0.10.0
 */
class PageBreak extends TextBreak
{
    /**
     * Write page break
     *
     * @since 0.12.0
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Writer\HTML $parentWriter Type hint */
        $parentWriter = $this->parentWriter;
        if ($parentWriter->isPdf()) {
            return '<pagebreak style="page-break-before: always;" pagebreak="true"></pagebreak>';
        }

        return '';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * Table element HTML writer
 *
 * @since 0.10.0
 */
class Table extends AbstractElement
{
    /**
     * Write table
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Table) {
            return '';
        }

        $content = '';
        $rows = $this->element->getRows();
        $rowCount = count($rows);
        if ($rowCount > 0) {
            $content .= '<table' . self::getTableStyle($this->element->getStyle()) . '>' . PHP_EOL;

            for ($i = 0; $i < $rowCount; $i++) {
                /** @var $row \PhpOffice\PhpWord\Element\Row Type hint */
                $rowStyle = $rows[$i]->getStyle();
                // $height = $row->getHeight();
                $tblHeader = $rowStyle->isTblHeader();
                $content .= '<tr>' . PHP_EOL;
                $rowCells = $rows[$i]->getCells();
                $rowCellCount = count($rowCells);
                for ($j = 0; $j < $rowCellCount; $j++) {
                    $cellStyle = $rowCells[$j]->getStyle();
                    $cellBgColor = $cellStyle->getBgColor();
                    $cellBgColor === 'auto' && $cellBgColor = null; // auto cannot be parsed to hexadecimal number
                    $cellFgColor = null;
                    if ($cellBgColor) {
                        $red = hexdec(substr($cellBgColor, 0, 2));
                        $green = hexdec(substr($cellBgColor, 2, 2));
                        $blue = hexdec(substr($cellBgColor, 4, 2));
                        $cellFgColor = (($red * 0.299 + $green * 0.587 + $blue * 0.114) > 186) ? null : 'ffffff';
                    }
                    $cellColSpan = $cellStyle->getGridSpan();
                    $cellRowSpan = 1;
                    $cellVMerge = $cellStyle->getVMerge();
                    // If this is the first cell of the vertical merge, find out how man rows it spans
                    if ($cellVMerge === 'restart') {
                        for ($k = $i + 1; $k < $rowCount; $k++) {
                            $kRowCells = $rows[$k]->getCells();
                            if (isset($kRowCells[$j])) {
                                if ($kRowCells[$j]->getStyle()->getVMerge() === 'continue') {
                                    $cellRowSpan++;
                                } else {
                                    break;
                                }
                            } else {
                                break;
                            }
                        }
                    }
                    // Ignore cells that are merged vertically with previous rows
                    if ($cellVMerge !== 'continue') {
                        $cellTag = $tblHeader ? 'th' : 'td';
                        $cellColSpanAttr = (is_numeric($cellColSpan) && ($cellColSpan > 1) ? " colspan=\"{$cellColSpan}\"" : '');
                        $cellRowSpanAttr = ($cellRowSpan > 1 ? " rowspan=\"{$cellRowSpan}\"" : '');
                        $cellBgColorAttr = (is_null($cellBgColor) ? '' : " bgcolor=\"#{$cellBgColor}\"");
                        $cellFgColorAttr = (is_null($cellFgColor) ? '' : " color=\"#{$cellFgColor}\"");
                        $content .= "<{$cellTag}{$cellColSpanAttr}{$cellRowSpanAttr}{$cellBgColorAttr}{$cellFgColorAttr}>" . PHP_EOL;
                        $writer = new Container($this->parentWriter, $rowCells[$j]);
                        $content .= $writer->write();
                        if ($cellRowSpan > 1) {
                            // There shouldn't be any content in the subsequent merged cells, but lets check anyway
                            for ($k = $i + 1; $k < $rowCount; $k++) {
                                $kRowCells = $rows[$k]->getCells();
                                if (isset($kRowCells[$j])) {
                                    if ($kRowCells[$j]->getStyle()->getVMerge() === 'continue') {
                                        $writer = new Container($this->parentWriter, $kRowCells[$j]);
                                        $content .= $writer->write();
                                    } else {
                                        break;
                                    }
                                } else {
                                    break;
                                }
                            }
                        }
                        $content .= "</{$cellTag}>" . PHP_EOL;
                    }
                }
                $content .= '</tr>' . PHP_EOL;
            }
            $content .= '</table>' . PHP_EOL;
        }

        return $content;
    }

    /**
     * Translates Table style in CSS equivalent
     *
     * @param string|\PhpOffice\PhpWord\Style\Table|null $tableStyle
     * @return string
     */
    private function getTableStyle($tableStyle = null)
    {
        if ($tableStyle == null) {
            return '';
        }
        if (is_string($tableStyle)) {
            $style = ' class="' . $tableStyle;
        } else {
            $style = ' style="';
            if ($tableStyle->getLayout() == \PhpOffice\PhpWord\Style\Table::LAYOUT_FIXED) {
                $style .= 'table-layout: fixed;';
            } elseif ($tableStyle->getLayout() == \PhpOffice\PhpWord\Style\Table::LAYOUT_AUTO) {
                $style .= 'table-layout: auto;';
            }
        }

        return $style . '"';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Element\TrackChange;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;
use PhpOffice\PhpWord\Writer\HTML\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\HTML\Style\Paragraph as ParagraphStyleWriter;

/**
 * Text element HTML writer
 *
 * @since 0.10.0
 */
class Text extends AbstractElement
{
    /**
     * Text written after opening
     *
     * @var string
     */
    private $openingText = '';

    /**
     * Text written before closing
     *
     * @var string
     */
    private $closingText = '';

    /**
     * Opening tags
     *
     * @var string
     */
    private $openingTags = '';

    /**
     * Closing tag
     *
     * @var string
     */
    private $closingTags = '';

    /**
     * Write text
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Element\Text $element Type hint */
        $element = $this->element;
        $this->getFontStyle();

        $content = '';
        $content .= $this->writeOpening();
        $content .= $this->openingText;
        $content .= $this->openingTags;
        if (Settings::isOutputEscapingEnabled()) {
            $content .= $this->escaper->escapeHtml($element->getText());
        } else {
            $content .= $element->getText();
        }
        $content .= $this->closingTags;
        $content .= $this->closingText;
        $content .= $this->writeClosing();

        return $content;
    }

    /**
     * Set opening text.
     *
     * @param string $value
     */
    public function setOpeningText($value)
    {
        $this->openingText = $value;
    }

    /**
     * Set closing text.
     *
     * @param string $value
     */
    public function setClosingText($value)
    {
        $this->closingText = $value;
    }

    /**
     * Write opening
     *
     * @return string
     */
    protected function writeOpening()
    {
        $content = '';
        if (!$this->withoutP) {
            $style = '';
            if (method_exists($this->element, 'getParagraphStyle')) {
                $style = $this->getParagraphStyle();
            }
            $content .= "<p{$style}>";
        }

        //open track change tag
        $content .= $this->writeTrackChangeOpening();

        return $content;
    }

    /**
     * Write ending
     *
     * @return string
     */
    protected function writeClosing()
    {
        $content = '';

        //close track change tag
        $content .= $this->writeTrackChangeClosing();

        if (!$this->withoutP) {
            if (Settings::isOutputEscapingEnabled()) {
                $content .= $this->escaper->escapeHtml($this->closingText);
            } else {
                $content .= $this->closingText;
            }

            $content .= '</p>' . PHP_EOL;
        }

        return $content;
    }

    /**
     * writes the track change opening tag
     *
     * @return string the HTML, an empty string if no track change information
     */
    private function writeTrackChangeOpening()
    {
        $changed = $this->element->getTrackChange();
        if ($changed == null) {
            return '';
        }

        $content = '';
        if (($changed->getChangeType() == TrackChange::INSERTED)) {
            $content .= '<ins data-phpword-prop=\'';
        } elseif ($changed->getChangeType() == TrackChange::DELETED) {
            $content .= '<del data-phpword-prop=\'';
        }

        $changedProp = array('changed' => array('author'=> $changed->getAuthor(), 'id'    => $this->element->getElementId()));
        if ($changed->getDate() != null) {
            $changedProp['changed']['date'] = $changed->getDate()->format('Y-m-d\TH:i:s\Z');
        }
        $content .= json_encode($changedProp);
        $content .= '\' ';
        $content .= 'title="' . $changed->getAuthor();
        if ($changed->getDate() != null) {
            $dateUser = $changed->getDate()->format('Y-m-d H:i:s');
            $content .= ' - ' . $dateUser;
        }
        $content .= '">';

        return $content;
    }

    /**
     * writes the track change closing tag
     *
     * @return string the HTML, an empty string if no track change information
     */
    private function writeTrackChangeClosing()
    {
        $changed = $this->element->getTrackChange();
        if ($changed == null) {
            return '';
        }

        $content = '';
        if (($changed->getChangeType() == TrackChange::INSERTED)) {
            $content .= '</ins>';
        } elseif ($changed->getChangeType() == TrackChange::DELETED) {
            $content .= '</del>';
        }

        return $content;
    }

    /**
     * Write paragraph style
     *
     * @return string
     */
    private function getParagraphStyle()
    {
        /** @var \PhpOffice\PhpWord\Element\Text $element Type hint */
        $element = $this->element;
        $style = '';
        if (!method_exists($element, 'getParagraphStyle')) {
            return $style;
        }

        $paragraphStyle = $element->getParagraphStyle();
        $pStyleIsObject = ($paragraphStyle instanceof Paragraph);
        if ($pStyleIsObject) {
            $styleWriter = new ParagraphStyleWriter($paragraphStyle);
            $style = $styleWriter->write();
        } elseif (is_string($paragraphStyle)) {
            $style = $paragraphStyle;
        }
        if ($style) {
            $attribute = $pStyleIsObject ? 'style' : 'class';
            $style = " {$attribute}=\"{$style}\"";
        }

        return $style;
    }

    /**
     * Get font style.
     */
    private function getFontStyle()
    {
        /** @var \PhpOffice\PhpWord\Element\Text $element Type hint */
        $element = $this->element;
        $style = '';
        $fontStyle = $element->getFontStyle();
        $fStyleIsObject = ($fontStyle instanceof Font);
        if ($fStyleIsObject) {
            $styleWriter = new FontStyleWriter($fontStyle);
            $style = $styleWriter->write();
        } elseif (is_string($fontStyle)) {
            $style = $fontStyle;
        }
        if ($style) {
            $attribute = $fStyleIsObject ? 'style' : 'class';
            $this->openingTags = "<span {$attribute}=\"{$style}\">";
            $this->closingTags = '</span>';
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/TextBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * TextBreak element HTML writer
 *
 * @since 0.10.0
 */
class TextBreak extends AbstractElement
{
    /**
     * Write text break
     *
     * @return string
     */
    public function write()
    {
        if ($this->withoutP) {
            $content = '<br />' . PHP_EOL;
        } else {
            $content = '<p>&nbsp;</p>' . PHP_EOL;
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/TextRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

/**
 * TextRun element HTML writer
 *
 * @since 0.10.0
 */
class TextRun extends Text
{
    /**
     * Write text run
     *
     * @return string
     */
    public function write()
    {
        $content = '';

        $content .= $this->writeOpening();
        $writer = new Container($this->parentWriter, $this->element);
        $content .= $writer->write();
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Element/Title.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Element;

use PhpOffice\PhpWord\Settings;

/**
 * TextRun element HTML writer
 *
 * @since 0.10.0
 */
class Title extends AbstractElement
{
    /**
     * Write heading
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Title) {
            return '';
        }

        $tag = 'h' . $this->element->getDepth();

        $text = $this->element->getText();
        if (is_string($text)) {
            if (Settings::isOutputEscapingEnabled()) {
                $text = $this->escaper->escapeHtml($text);
            }
        } elseif ($text instanceof \PhpOffice\PhpWord\Element\AbstractContainer) {
            $writer = new Container($this->parentWriter, $text);
            $text = $writer->write();
        }

        $content = "<{$tag}>{$text}</{$tag}>" . PHP_EOL;

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Part/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Part;

use Laminas\Escaper\Escaper;
use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Writer\AbstractWriter;

/**
 * @since 0.11.0
 */
abstract class AbstractPart
{
    /**
     * @var \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    private $parentWriter;

    /**
     * @var \Laminas\Escaper\Escaper
     */
    protected $escaper;

    public function __construct()
    {
        $this->escaper = new Escaper();
    }

    /**
     * @return string
     */
    abstract public function write();

    /**
     * @param \PhpOffice\PhpWord\Writer\AbstractWriter $writer
     */
    public function setParentWriter(AbstractWriter $writer = null)
    {
        $this->parentWriter = $writer;
    }

    /**
     * @throws \PhpOffice\PhpWord\Exception\Exception
     *
     * @return \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    public function getParentWriter()
    {
        if ($this->parentWriter !== null) {
            return $this->parentWriter;
        }
        throw new Exception('No parent WriterInterface assigned.');
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Part/Body.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Part;

use PhpOffice\PhpWord\Writer\HTML\Element\Container;
use PhpOffice\PhpWord\Writer\HTML\Element\TextRun as TextRunWriter;

/**
 * RTF body part writer
 *
 * @since 0.11.0
 */
class Body extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();

        $content = '';

        $content .= '<body>' . PHP_EOL;
        $sections = $phpWord->getSections();
        foreach ($sections as $section) {
            $writer = new Container($this->getParentWriter(), $section);
            $content .= $writer->write();
        }

        $content .= $this->writeNotes();
        $content .= '</body>' . PHP_EOL;

        return $content;
    }

    /**
     * Write footnote/endnote contents as textruns
     *
     * @return string
     */
    private function writeNotes()
    {
        /** @var \PhpOffice\PhpWord\Writer\HTML $parentWriter Type hint */
        $parentWriter = $this->getParentWriter();
        $phpWord = $parentWriter->getPhpWord();
        $notes = $parentWriter->getNotes();

        $content = '';

        if (!empty($notes)) {
            $content .= '<hr />' . PHP_EOL;
            foreach ($notes as $noteId => $noteMark) {
                list($noteType, $noteTypeId) = explode('-', $noteMark);
                $method = 'get' . ($noteType == 'endnote' ? 'Endnotes' : 'Footnotes');
                $collection = $phpWord->$method()->getItems();

                if (isset($collection[$noteTypeId])) {
                    $element = $collection[$noteTypeId];
                    $noteAnchor = "<a name=\"note-{$noteId}\" />";
                    $noteAnchor .= "<a href=\"#{$noteMark}\" class=\"NoteRef\"><sup>{$noteId}</sup></a>";

                    $writer = new TextRunWriter($this->getParentWriter(), $element);
                    $writer->setOpeningText($noteAnchor);
                    $content .= $writer->write();
                }
            }
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Part/Head.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Part;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;
use PhpOffice\PhpWord\Writer\HTML\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\HTML\Style\Generic as GenericStyleWriter;
use PhpOffice\PhpWord\Writer\HTML\Style\Paragraph as ParagraphStyleWriter;

/**
 * RTF head part writer
 *
 * @since 0.11.0
 */
class Head extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $docProps = $this->getParentWriter()->getPhpWord()->getDocInfo();
        $propertiesMapping = array(
            'creator'     => 'author',
            'title'       => '',
            'description' => '',
            'subject'     => '',
            'keywords'    => '',
            'category'    => '',
            'company'     => '',
            'manager'     => '',
        );
        $title = $docProps->getTitle();
        $title = ($title != '') ? $title : 'PHPWord';

        $content = '';

        $content .= '<head>' . PHP_EOL;
        $content .= '<meta charset="UTF-8" />' . PHP_EOL;
        $content .= '<title>' . $title . '</title>' . PHP_EOL;
        foreach ($propertiesMapping as $key => $value) {
            $value = ($value == '') ? $key : $value;
            $method = 'get' . $key;
            if ($docProps->$method() != '') {
                $content .= '<meta name="' . $value . '"'
                          . ' content="' . (Settings::isOutputEscapingEnabled() ? $this->escaper->escapeHtmlAttr($docProps->$method()) : $docProps->$method()) . '"'
                          . ' />' . PHP_EOL;
            }
        }
        $content .= $this->writeStyles();
        $content .= '</head>' . PHP_EOL;

        return $content;
    }

    /**
     * Get styles
     *
     * @return string
     */
    private function writeStyles()
    {
        $css = '<style>' . PHP_EOL;

        // Default styles
        $defaultStyles = array(
            '*' => array(
                'font-family' => Settings::getDefaultFontName(),
                'font-size'   => Settings::getDefaultFontSize() . 'pt',
            ),
            'a.NoteRef' => array(
                'text-decoration' => 'none',
            ),
            'hr' => array(
                'height'     => '1px',
                'padding'    => '0',
                'margin'     => '1em 0',
                'border'     => '0',
                'border-top' => '1px solid #CCC',
            ),
            'table' => array(
                'border'         => '1px solid black',
                'border-spacing' => '0px',
                'width '         => '100%',
            ),
            'td' => array(
                'border' => '1px solid black',
            ),
        );
        foreach ($defaultStyles as $selector => $style) {
            $styleWriter = new GenericStyleWriter($style);
            $css .= $selector . ' {' . $styleWriter->write() . '}' . PHP_EOL;
        }

        // Custom styles
        $customStyles = Style::getStyles();
        if (is_array($customStyles)) {
            foreach ($customStyles as $name => $style) {
                if ($style instanceof Font) {
                    $styleWriter = new FontStyleWriter($style);
                    if ($style->getStyleType() == 'title') {
                        $name = str_replace('Heading_', 'h', $name);
                    } else {
                        $name = '.' . $name;
                    }
                    $css .= "{$name} {" . $styleWriter->write() . '}' . PHP_EOL;
                } elseif ($style instanceof Paragraph) {
                    $styleWriter = new ParagraphStyleWriter($style);
                    $name = '.' . $name;
                    $css .= "{$name} {" . $styleWriter->write() . '}' . PHP_EOL;
                }
            }
        }
        $css .= '</style>' . PHP_EOL;

        return $css;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Style/AbstractStyle.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Style;

use PhpOffice\PhpWord\Style\AbstractStyle as Style;

/**
 * Style writer
 *
 * @since 0.10.0
 */
abstract class AbstractStyle
{
    /**
     * Parent writer
     *
     * @var \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    private $parentWriter;

    /**
     * Style
     *
     * @var array|\PhpOffice\PhpWord\Style\AbstractStyle
     */
    private $style;

    /**
     * Write style
     */
    abstract public function write();

    /**
     * Create new instance
     *
     * @param array|\PhpOffice\PhpWord\Style\AbstractStyle $style
     */
    public function __construct($style = null)
    {
        $this->style = $style;
    }

    /**
     * Set parent writer.
     *
     * @param \PhpOffice\PhpWord\Writer\AbstractWriter $writer
     */
    public function setParentWriter($writer)
    {
        $this->parentWriter = $writer;
    }

    /**
     * Get parent writer
     *
     * @return \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    public function getParentWriter()
    {
        return $this->parentWriter;
    }

    /**
     * Get style
     *
     * @return array|\PhpOffice\PhpWord\Style\AbstractStyle $style
     */
    public function getStyle()
    {
        if (!$this->style instanceof Style && !is_array($this->style)) {
            return '';
        }

        return $this->style;
    }

    /**
     * Takes array where of CSS properties / values and converts to CSS string
     *
     * @param array $css
     * @return string
     */
    protected function assembleCss($css)
    {
        $pairs = array();
        $string = '';
        foreach ($css as $key => $value) {
            if ($value != '') {
                $pairs[] = $key . ': ' . $value;
            }
        }
        if (!empty($pairs)) {
            $string = implode('; ', $pairs) . ';';
        }

        return $string;
    }

    /**
     * Get value if ...
     *
     * @param bool|null $condition
     * @param string $value
     * @return string
     */
    protected function getValueIf($condition, $value)
    {
        return $condition == true ? $value : '';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Style/Font.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Style;

use PhpOffice\PhpWord\Style\Font as FontStyle;

/**
 * Font style HTML writer
 *
 * @since 0.10.0
 */
class Font extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof FontStyle) {
            return '';
        }
        $css = array();

        $font = $style->getName();
        $size = $style->getSize();
        $color = $style->getColor();
        $fgColor = $style->getFgColor();
        $underline = $style->getUnderline() != FontStyle::UNDERLINE_NONE;
        $lineThrough = $style->isStrikethrough() || $style->isDoubleStrikethrough();

        $css['font-family'] = $this->getValueIf($font !== null, "'{$font}'");
        $css['font-size'] = $this->getValueIf($size !== null, "{$size}pt");
        $css['color'] = $this->getValueIf($color !== null, "#{$color}");
        $css['background'] = $this->getValueIf($fgColor != '', $fgColor);
        $css['font-weight'] = $this->getValueIf($style->isBold(), 'bold');
        $css['font-style'] = $this->getValueIf($style->isItalic(), 'italic');
        $css['vertical-align'] = '';
        $css['vertical-align'] .= $this->getValueIf($style->isSuperScript(), 'super');
        $css['vertical-align'] .= $this->getValueIf($style->isSubScript(), 'sub');
        $css['text-decoration'] = '';
        $css['text-decoration'] .= $this->getValueIf($underline, 'underline ');
        $css['text-decoration'] .= $this->getValueIf($lineThrough, 'line-through ');
        $css['text-transform'] = $this->getValueIf($style->isAllCaps(), 'uppercase');
        $css['font-variant'] = $this->getValueIf($style->isSmallCaps(), 'small-caps');
        $css['display'] = $this->getValueIf($style->isHidden(), 'none');

        $spacing = $style->getSpacing();
        $css['letter-spacing'] = $this->getValueIf(!is_null($spacing), ($spacing / 20) . 'pt');

        return $this->assembleCss($css);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Style/Generic.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Style;

/**
 * Generic style writer
 *
 * @since 0.10.0
 */
class Generic extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        $css = array();

        if (is_array($style) && !empty($style)) {
            $css = $style;
        }

        return $this->assembleCss($css);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Style/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Style;

/**
 * Paragraph style HTML writer
 *
 * @since 0.10.0
 */
class Image extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Image) {
            return '';
        }
        $css = array();

        $width = $style->getWidth();
        $height = $style->getHeight();
        $css['width'] = $this->getValueIf(is_numeric($width), $width . 'px');
        $css['height'] = $this->getValueIf(is_numeric($height), $height . 'px');

        return $this->assembleCss($css);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML/Style/Paragraph.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\HTML\Style;

use PhpOffice\PhpWord\SimpleType\Jc;

/**
 * Paragraph style HTML writer
 *
 * @since 0.10.0
 */
class Paragraph extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Paragraph) {
            return '';
        }
        $css = array();

        // Alignment
        if ('' !== $style->getAlignment()) {
            $textAlign = '';

            switch ($style->getAlignment()) {
                case Jc::CENTER:
                    $textAlign = 'center';
                    break;
                case Jc::END:
                case Jc::MEDIUM_KASHIDA:
                case Jc::HIGH_KASHIDA:
                case Jc::LOW_KASHIDA:
                case Jc::RIGHT:
                    $textAlign = 'right';
                    break;
                case Jc::BOTH:
                case Jc::DISTRIBUTE:
                case Jc::THAI_DISTRIBUTE:
                case Jc::JUSTIFY:
                    $textAlign = 'justify';
                    break;
                default: //all others, align left
                    $textAlign = 'left';
                    break;
            }

            $css['text-align'] = $textAlign;
        }

        // Spacing
        $spacing = $style->getSpace();
        if (!is_null($spacing)) {
            $before = $spacing->getBefore();
            $after = $spacing->getAfter();
            $css['margin-top'] = $this->getValueIf(!is_null($before), ($before / 20) . 'pt');
            $css['margin-bottom'] = $this->getValueIf(!is_null($after), ($after / 20) . 'pt');
        } else {
            $css['margin-top'] = '0';
            $css['margin-bottom'] = '0';
        }

        return $this->assembleCss($css);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/HTML.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\PhpWord;

/**
 * HTML writer
 *
 * Not supported: PreserveText, PageBreak, Object
 * @since 0.10.0
 */
class HTML extends AbstractWriter implements WriterInterface
{
    /**
     * Is the current writer creating PDF?
     *
     * @var bool
     */
    protected $isPdf = false;

    /**
     * Footnotes and endnotes collection
     *
     * @var array
     */
    protected $notes = array();

    /**
     * Create new instance
     */
    public function __construct(PhpWord $phpWord = null)
    {
        $this->setPhpWord($phpWord);

        $this->parts = array('Head', 'Body');
        foreach ($this->parts as $partName) {
            $partClass = 'PhpOffice\\PhpWord\\Writer\\HTML\\Part\\' . $partName;
            if (class_exists($partClass)) {
                /** @var \PhpOffice\PhpWord\Writer\HTML\Part\AbstractPart $part Type hint */
                $part = new $partClass();
                $part->setParentWriter($this);
                $this->writerParts[strtolower($partName)] = $part;
            }
        }
    }

    /**
     * Save PhpWord to file.
     *
     * @param string $filename
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function save($filename = null)
    {
        $this->writeFile($this->openFile($filename), $this->getContent());
    }

    /**
     * Get content
     *
     * @return string
     * @since 0.11.0
     */
    public function getContent()
    {
        $content = '';

        $content .= '<!DOCTYPE html>' . PHP_EOL;
        $content .= '<!-- Generated by PHPWord -->' . PHP_EOL;
        $content .= '<html>' . PHP_EOL;
        $content .= $this->getWriterPart('Head')->write();
        $content .= $this->getWriterPart('Body')->write();
        $content .= '</html>' . PHP_EOL;

        return $content;
    }

    /**
     * Get is PDF
     *
     * @return bool
     */
    public function isPdf()
    {
        return $this->isPdf;
    }

    /**
     * Get notes
     *
     * @return array
     */
    public function getNotes()
    {
        return $this->notes;
    }

    /**
     * Add note.
     *
     * @param int $noteId
     * @param string $noteMark
     */
    public function addNote($noteId, $noteMark)
    {
        $this->notes[$noteId] = $noteMark;
    }

    /**
     * Write document
     *
     * @deprecated 0.11.0
     *
     * @return string
     *
     * @codeCoverageIgnore
     */
    public function writeDocument()
    {
        return $this->getContent();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/AbstractElement.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

use PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement as Word2007AbstractElement;

/**
 * Abstract element writer
 *
 * @since 0.11.0
 */
abstract class AbstractElement extends Word2007AbstractElement
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Container.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

use PhpOffice\PhpWord\Writer\Word2007\Element\Container as Word2007Container;

/**
 * Container element writer (section, textrun, header, footnote, cell, etc.)
 *
 * @since 0.11.0
 */
class Container extends Word2007Container
{
    /**
     * Namespace; Can't use __NAMESPACE__ in inherited class (ODText)
     *
     * @var string
     */
    protected $namespace = 'PhpOffice\\PhpWord\\Writer\\ODText\\Element';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Field.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */
// Not fully implemented
//     - supports only PAGE and NUMPAGES
//     - supports only default formats and options
//     - supports style only if specified by name
//     - spaces before and after field may be dropped

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * Field element writer
 *
 * @since 0.11.0
 */
class Field extends Text
{
    /**
     * Write field element.
     */
    public function write()
    {
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Field) {
            return;
        }

        $type = strtolower($element->getType());
        switch ($type) {
            case 'date':
            case 'page':
            case 'numpages':
                $this->writeDefault($element, $type);
                break;
        }
    }

    private function writeDefault(\PhpOffice\PhpWord\Element\Field $element, $type)
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('text:span');
        if (method_exists($element, 'getFontStyle')) {
            $fstyle = $element->getFontStyle();
            if (is_string($fstyle)) {
                $xmlWriter->writeAttribute('text:style-name', $fstyle);
            }
        }
        switch ($type) {
            case 'date':
                $xmlWriter->startElement('text:date');
                $xmlWriter->writeAttribute('text:fixed', 'false');
                $xmlWriter->endElement();
                break;
            case 'page':
                $xmlWriter->startElement('text:page-number');
                $xmlWriter->writeAttribute('text:fixed', 'false');
                $xmlWriter->endElement();
                break;
            case 'numpages':
                $xmlWriter->startElement('text:page-count');
                $xmlWriter->endElement();
                break;
        }
        $xmlWriter->endElement(); // text:span
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

use PhpOffice\PhpWord\Shared\Converter;

/**
 * Image element writer
 *
 * @since 0.10.0
 */
class Image extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Image) {
            return;
        }

        $mediaIndex = $element->getMediaIndex();
        $target = 'Pictures/' . $element->getTarget();
        $style = $element->getStyle();
        $width = Converter::pixelToCm($style->getWidth());
        $height = Converter::pixelToCm($style->getHeight());

        $xmlWriter->startElement('text:p');
        $xmlWriter->writeAttribute('text:style-name', 'IM' . $mediaIndex);

        $xmlWriter->startElement('draw:frame');
        $xmlWriter->writeAttribute('draw:style-name', 'fr' . $mediaIndex);
        $xmlWriter->writeAttribute('draw:name', $element->getElementId());
        $xmlWriter->writeAttribute('text:anchor-type', 'as-char');
        $xmlWriter->writeAttribute('svg:width', $width . 'cm');
        $xmlWriter->writeAttribute('svg:height', $height . 'cm');
        $xmlWriter->writeAttribute('draw:z-index', $mediaIndex);

        $xmlWriter->startElement('draw:image');
        $xmlWriter->writeAttribute('xlink:href', $target);
        $xmlWriter->writeAttribute('xlink:type', 'simple');
        $xmlWriter->writeAttribute('xlink:show', 'embed');
        $xmlWriter->writeAttribute('xlink:actuate', 'onLoad');
        $xmlWriter->endElement(); // draw:image

        $xmlWriter->endElement(); // draw:frame

        $xmlWriter->endElement(); // text:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Link.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * Text element writer
 *
 * @since 0.10.0
 */
class Link extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Link) {
            return;
        }

        if (!$this->withoutP) {
            $xmlWriter->startElement('text:p'); // text:p
        }

        $xmlWriter->startElement('text:a');
        $xmlWriter->writeAttribute('xlink:type', 'simple');
        $xmlWriter->writeAttribute('xlink:href', ($element->isInternal() ? '#' : '') . $element->getSource());
        $this->writeText($element->getText());
        $xmlWriter->endElement(); // text:a

        if (!$this->withoutP) {
            $xmlWriter->endElement(); // text:p
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/PageBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * PageBreak element writer
 */
class PageBreak extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('text:p');
        $xmlWriter->writeAttribute('text:style-name', 'PB');
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

use PhpOffice\PhpWord\Element\Row as RowElement;
use PhpOffice\PhpWord\Element\Table as TableElement;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Table element writer
 *
 * @since 0.10.0
 */
class Table extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Table) {
            return;
        }
        $rows = $element->getRows();
        $rowCount = count($rows);

        if ($rowCount > 0) {
            $xmlWriter->startElement('table:table');
            $xmlWriter->writeAttribute('table:name', $element->getElementId());
            $xmlWriter->writeAttribute('table:style', $element->getElementId());

            // Write columns
            $this->writeColumns($xmlWriter, $element);

            // Write rows
            foreach ($rows as $row) {
                $this->writeRow($xmlWriter, $row);
            }
            $xmlWriter->endElement(); // table:table
        }
    }

    /**
     * Write column.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Table $element
     */
    private function writeColumns(XMLWriter $xmlWriter, TableElement $element)
    {
        $colCount = $element->countColumns();

        for ($i = 0; $i < $colCount; $i++) {
            $xmlWriter->startElement('table:table-column');
            $xmlWriter->writeAttribute('table:style-name', $element->getElementId() . '.' . $i);
            $xmlWriter->endElement();
        }
    }

    /**
     * Write row.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Row $row
     */
    private function writeRow(XMLWriter $xmlWriter, RowElement $row)
    {
        $xmlWriter->startElement('table:table-row');
        /** @var $row \PhpOffice\PhpWord\Element\Row Type hint */
        foreach ($row->getCells() as $cell) {
            $xmlWriter->startElement('table:table-cell');
            $xmlWriter->writeAttribute('office:value-type', 'string');

            $containerWriter = new Container($xmlWriter, $cell);
            $containerWriter->write();

            $xmlWriter->endElement(); // table:table-cell
        }
        $xmlWriter->endElement(); // table:table-row
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

use PhpOffice\PhpWord\Element\TrackChange;
use PhpOffice\PhpWord\Exception\Exception;

/**
 * Text element writer
 *
 * @since 0.10.0
 */
class Text extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Text) {
            return;
        }
        $fontStyle = $element->getFontStyle();
        $paragraphStyle = $element->getParagraphStyle();

        // @todo Commented for TextRun. Should really checkout this value
        // $fStyleIsObject = ($fontStyle instanceof Font) ? true : false;
        //$fStyleIsObject = false;

        //if ($fStyleIsObject) {
        // Don't never be the case, because I browse all sections for cleaning all styles not declared
        //    throw new Exception('PhpWord : $fStyleIsObject wouldn\'t be an object');
        //}

        if (!$this->withoutP) {
            $xmlWriter->startElement('text:p'); // text:p
        }
        if ($element->getTrackChange() != null && $element->getTrackChange()->getChangeType() == TrackChange::DELETED) {
            $xmlWriter->startElement('text:change');
            $xmlWriter->writeAttribute('text:change-id', $element->getTrackChange()->getElementId());
            $xmlWriter->endElement();
        } else {
            if (empty($fontStyle)) {
                if (empty($paragraphStyle)) {
                    if (!$this->withoutP) {
                        $xmlWriter->writeAttribute('text:style-name', 'Normal');
                    }
                } elseif (is_string($paragraphStyle)) {
                    if (!$this->withoutP) {
                        $xmlWriter->writeAttribute('text:style-name', $paragraphStyle);
                    }
                }
                $this->writeChangeInsertion(true, $element->getTrackChange());
                $this->replaceTabs($element->getText(), $xmlWriter);
                $this->writeChangeInsertion(false, $element->getTrackChange());
            } else {
                if (empty($paragraphStyle)) {
                    if (!$this->withoutP) {
                        $xmlWriter->writeAttribute('text:style-name', 'Normal');
                    }
                } elseif (is_string($paragraphStyle)) {
                    if (!$this->withoutP) {
                        $xmlWriter->writeAttribute('text:style-name', $paragraphStyle);
                    }
                }
                // text:span
                $xmlWriter->startElement('text:span');
                if (is_string($fontStyle)) {
                    $xmlWriter->writeAttribute('text:style-name', $fontStyle);
                }
                $this->writeChangeInsertion(true, $element->getTrackChange());
                $this->replaceTabs($element->getText(), $xmlWriter);
                $this->writeChangeInsertion(false, $element->getTrackChange());
                $xmlWriter->endElement();
            }
        }
        if (!$this->withoutP) {
            $xmlWriter->endElement(); // text:p
        }
    }

    private function replacetabs($text, $xmlWriter)
    {
        if (preg_match('/^ +/', $text, $matches)) {
            $num = strlen($matches[0]);
            $xmlWriter->startElement('text:s');
            $xmlWriter->writeAttributeIf($num > 1, 'text:c', "$num");
            $xmlWriter->endElement();
            $text = preg_replace('/^ +/', '', $text);
        }
        preg_match_all('/([\\s\\S]*?)(\\t|  +| ?$)/', $text, $matches, PREG_SET_ORDER);
        foreach ($matches as $match) {
            $this->writeText($match[1]);
            if ($match[2] === '') {
                break;
            } elseif ($match[2] === "\t") {
                $xmlWriter->writeElement('text:tab');
            } elseif ($match[2] === ' ') {
                $xmlWriter->writeElement('text:s');
                break;
            } else {
                $num = strlen($match[2]);
                $xmlWriter->startElement('text:s');
                $xmlWriter->writeAttributeIf($num > 1, 'text:c', "$num");
                $xmlWriter->endElement();
            }
        }
    }

    private function writeChangeInsertion($start = true, TrackChange $trackChange = null)
    {
        if ($trackChange == null || $trackChange->getChangeType() != TrackChange::INSERTED) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('text:change-' . ($start ? 'start' : 'end'));
        $xmlWriter->writeAttribute('text:change-id', $trackChange->getElementId());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/TextBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * TextBreak element writer
 *
 * @since 0.10.0
 */
class TextBreak extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('text:p');
        $xmlWriter->writeAttribute('text:style-name', 'Standard');
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/TextRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * TextRun element writer
 *
 * @since 0.10.0
 */
class TextRun extends Text
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();

        $xmlWriter->startElement('text:p');
        /** @scrutinizer ignore-call */
        $pStyle = $element->getParagraphStyle();
        if (!is_string($pStyle)) {
            $pStyle = 'Normal';
        }
        $xmlWriter->writeAttribute('text:style-name', $pStyle);

        $containerWriter = new Container($xmlWriter, $element);
        $containerWriter->write();

        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Element/Title.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Element;

/**
 * Title element writer
 *
 * @since 0.11.0
 */
class Title extends AbstractElement
{
    /**
     * Write element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Title) {
            return;
        }

        $xmlWriter->startElement('text:h');
        $hdname = 'HD';
        $sect = $element->getParent();
        if ($sect instanceof \PhpOffice\PhpWord\Element\Section) {
            if (self::compareToFirstElement($element, $sect->getElements())) {
                $hdname = 'HE';
            }
        }
        $depth = $element->getDepth();
        $xmlWriter->writeAttribute('text:style-name', "$hdname$depth");
        $xmlWriter->writeAttribute('text:outline-level', $depth);
        $xmlWriter->startElement('text:span');
        if ($depth > 0) {
            $xmlWriter->writeAttribute('text:style-name', 'Heading_' . $depth);
        } else {
            $xmlWriter->writeAttribute('text:style-name', 'Title');
        }
        $text = $element->getText();
        if (is_string($text)) {
            $this->writeText($text);
        } elseif ($text instanceof \PhpOffice\PhpWord\Element\AbstractContainer) {
            $containerWriter = new Container($xmlWriter, $text);
            $containerWriter->write();
        }
        $xmlWriter->endElement(); // text:span
        $xmlWriter->endElement(); // text:h
    }

    /**
     * Test if element is same as first element in array
     *
     * @param \PhpOffice\PhpWord\Element\AbstractElement $elem
     *
     * @param \PhpOffice\PhpWord\Element\AbstractElement[] $elemarray
     *
     * @return bool
     */
    private static function compareToFirstElement($elem, $elemarray)
    {
        return $elem === $elemarray[0];
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Writer\Word2007\Part\AbstractPart as Word2007AbstractPart;

/**
 * ODText writer part abstract
 */
abstract class AbstractPart extends Word2007AbstractPart
{
    /**
     * @var string Date format
     */
    protected $dateFormat = 'Y-m-d\TH:i:s.000';

    /**
     * Write common root attributes.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    protected function writeCommonRootAttributes(XMLWriter $xmlWriter)
    {
        $xmlWriter->writeAttribute('office:version', '1.2');
        $xmlWriter->writeAttribute('xmlns:office', 'urn:oasis:names:tc:opendocument:xmlns:office:1.0');
        $xmlWriter->writeAttribute('xmlns:style', 'urn:oasis:names:tc:opendocument:xmlns:style:1.0');
        $xmlWriter->writeAttribute('xmlns:text', 'urn:oasis:names:tc:opendocument:xmlns:text:1.0');
        $xmlWriter->writeAttribute('xmlns:table', 'urn:oasis:names:tc:opendocument:xmlns:table:1.0');
        $xmlWriter->writeAttribute('xmlns:draw', 'urn:oasis:names:tc:opendocument:xmlns:drawing:1.0');
        $xmlWriter->writeAttribute('xmlns:fo', 'urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0');
        $xmlWriter->writeAttribute('xmlns:xlink', 'http://www.w3.org/1999/xlink');
        $xmlWriter->writeAttribute('xmlns:dc', 'http://purl.org/dc/elements/1.1/');
        $xmlWriter->writeAttribute('xmlns:meta', 'urn:oasis:names:tc:opendocument:xmlns:meta:1.0');
        $xmlWriter->writeAttribute('xmlns:number', 'urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0');
        $xmlWriter->writeAttribute('xmlns:svg', 'urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0');
        $xmlWriter->writeAttribute('xmlns:chart', 'urn:oasis:names:tc:opendocument:xmlns:chart:1.0');
        $xmlWriter->writeAttribute('xmlns:dr3d', 'urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0');
        $xmlWriter->writeAttribute('xmlns:math', 'http://www.w3.org/1998/Math/MathML');
        $xmlWriter->writeAttribute('xmlns:form', 'urn:oasis:names:tc:opendocument:xmlns:form:1.0');
        $xmlWriter->writeAttribute('xmlns:script', 'urn:oasis:names:tc:opendocument:xmlns:script:1.0');
        $xmlWriter->writeAttribute('xmlns:ooo', 'http://openoffice.org/2004/office');
        $xmlWriter->writeAttribute('xmlns:ooow', 'http://openoffice.org/2004/writer');
        $xmlWriter->writeAttribute('xmlns:oooc', 'http://openoffice.org/2004/calc');
        $xmlWriter->writeAttribute('xmlns:dom', 'http://www.w3.org/2001/xml-events');
        $xmlWriter->writeAttribute('xmlns:rpt', 'http://openoffice.org/2005/report');
        $xmlWriter->writeAttribute('xmlns:of', 'urn:oasis:names:tc:opendocument:xmlns:of:1.2');
        $xmlWriter->writeAttribute('xmlns:xhtml', 'http://www.w3.org/1999/xhtml');
        $xmlWriter->writeAttribute('xmlns:grddl', 'http://www.w3.org/2003/g/data-view#');
        $xmlWriter->writeAttribute('xmlns:tableooo', 'http://openoffice.org/2009/table');
        $xmlWriter->writeAttribute('xmlns:css3t', 'http://www.w3.org/TR/css3-text/');
    }

    /**
     * Write font faces declaration.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    protected function writeFontFaces(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('office:font-face-decls');
        $fontTable = array();
        $styles = Style::getStyles();
        $numFonts = 0;
        if (count($styles) > 0) {
            foreach ($styles as $style) {
                // Font
                if ($style instanceof Font) {
                    $numFonts++;
                    $name = $style->getName();
                    if (!in_array($name, $fontTable)) {
                        $fontTable[] = $name;

                        // style:font-face
                        $xmlWriter->startElement('style:font-face');
                        $xmlWriter->writeAttribute('style:name', $name);
                        $xmlWriter->writeAttribute('svg:font-family', $name);
                        $xmlWriter->endElement();
                    }
                }
            }
        }
        if (!in_array(Settings::getDefaultFontName(), $fontTable)) {
            $xmlWriter->startElement('style:font-face');
            $xmlWriter->writeAttribute('style:name', Settings::getDefaultFontName());
            $xmlWriter->writeAttribute('svg:font-family', Settings::getDefaultFontName());
            $xmlWriter->endElement();
        }
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/Content.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

use PhpOffice\PhpWord\Element\AbstractContainer;
use PhpOffice\PhpWord\Element\Field;
use PhpOffice\PhpWord\Element\Image;
use PhpOffice\PhpWord\Element\Table;
use PhpOffice\PhpWord\Element\Text;
use PhpOffice\PhpWord\Element\TextRun;
use PhpOffice\PhpWord\Element\TrackChange;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Style\Paragraph;
use PhpOffice\PhpWord\Style\Table as TableStyle;
use PhpOffice\PhpWord\Writer\ODText\Element\Container;
use PhpOffice\PhpWord\Writer\ODText\Style\Paragraph as ParagraphStyleWriter;

/**
 * ODText content part writer: content.xml
 */
class Content extends AbstractPart
{
    /**
     * Auto style collection
     *
     * Collect inline style information from section, image, and table elements
     *
     * @todo Merge font and paragraph styles
     * @var array
     */
    private $autoStyles = array('Section' => array(), 'Image' => array(), 'Table' => array());
    private $imageParagraphStyles = array();

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $phpWord = $this->getParentWriter()->getPhpWord();

        $this->getAutoStyles($phpWord);

        $xmlWriter->startDocument('1.0', 'UTF-8');
        $xmlWriter->startElement('office:document-content');
        $this->writeCommonRootAttributes($xmlWriter);
        $xmlWriter->writeAttribute('xmlns:xforms', 'http://www.w3.org/2002/xforms');
        $xmlWriter->writeAttribute('xmlns:xsd', 'http://www.w3.org/2001/XMLSchema');
        $xmlWriter->writeAttribute('xmlns:xsi', 'http://www.w3.org/2001/XMLSchema-instance');
        $xmlWriter->writeAttribute('xmlns:field', 'urn:openoffice:names:experimental:ooo-ms-interop:xmlns:field:1.0');
        $xmlWriter->writeAttribute('xmlns:formx', 'urn:openoffice:names:experimental:ooxml-odf-interop:xmlns:form:1.0');

        // Font declarations and automatic styles
        $this->writeFontFaces($xmlWriter); // office:font-face-decls
        $this->writeAutoStyles($xmlWriter); // office:automatic-styles

        // Body
        $xmlWriter->startElement('office:body');
        $xmlWriter->startElement('office:text');

        // Tracked changes declarations
        $trackedChanges = array();
        $sections = $phpWord->getSections();
        foreach ($sections as $section) {
            $this->collectTrackedChanges($section, $trackedChanges);
        }
        $xmlWriter->startElement('text:tracked-changes');
        foreach ($trackedChanges as $trackedElement) {
            $trackedChange = $trackedElement->getTrackChange();
            $xmlWriter->startElement('text:changed-region');
            $trackedChange->setElementId();
            $xmlWriter->writeAttribute('text:id', $trackedChange->getElementId());

            if (($trackedChange->getChangeType() == TrackChange::INSERTED)) {
                $xmlWriter->startElement('text:insertion');
            } elseif ($trackedChange->getChangeType() == TrackChange::DELETED) {
                $xmlWriter->startElement('text:deletion');
            }

            $xmlWriter->startElement('office:change-info');
            $xmlWriter->writeElement('dc:creator', $trackedChange->getAuthor());
            if ($trackedChange->getDate() != null) {
                $xmlWriter->writeElement('dc:date', $trackedChange->getDate()->format('Y-m-d\TH:i:s\Z'));
            }
            $xmlWriter->endElement(); // office:change-info
            if ($trackedChange->getChangeType() == TrackChange::DELETED) {
                $xmlWriter->writeElement('text:p', $trackedElement->getText());
            }

            $xmlWriter->endElement(); // text:insertion|text:deletion
            $xmlWriter->endElement(); // text:changed-region
        }
        $xmlWriter->endElement(); // text:tracked-changes

        // Sequence declarations
        $sequences = array('Illustration', 'Table', 'Text', 'Drawing');
        $xmlWriter->startElement('text:sequence-decls');
        foreach ($sequences as $sequence) {
            $xmlWriter->startElement('text:sequence-decl');
            $xmlWriter->writeAttribute('text:display-outline-level', 0);
            $xmlWriter->writeAttribute('text:name', $sequence);
            $xmlWriter->endElement();
        }
        $xmlWriter->endElement(); // text:sequence-decl

        // Sections
        $sections = $phpWord->getSections();
        foreach ($sections as $section) {
            $name = 'Section' . $section->getSectionId();
            $xmlWriter->startElement('text:section');
            $xmlWriter->writeAttribute('text:name', $name);
            $xmlWriter->writeAttribute('text:style-name', $name);
            $xmlWriter->startElement('text:p');
            $xmlWriter->writeAttribute('text:style-name', 'SB' . $section->getSectionId());
            $xmlWriter->endElement();
            $containerWriter = new Container($xmlWriter, $section);
            $containerWriter->write();
            $xmlWriter->endElement(); // text:section
        }

        $xmlWriter->endElement(); // office:text
        $xmlWriter->endElement(); // office:body

        $xmlWriter->endElement(); // office:document-content

        return $xmlWriter->getData();
    }

    /**
     * Write automatic styles other than fonts and paragraphs.
     *
     * @since 0.11.0
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeAutoStyles(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('office:automatic-styles');

        $this->writeTextStyles($xmlWriter);
        foreach ($this->autoStyles as $element => $styles) {
            $writerClass = 'PhpOffice\\PhpWord\\Writer\\ODText\\Style\\' . $element;
            foreach ($styles as $style) {
                /** @var \PhpOffice\PhpWord\Writer\ODText\Style\AbstractStyle $styleWriter Type hint */
                $styleWriter = new $writerClass($xmlWriter, $style);
                $styleWriter->write();
            }
        }

        $xmlWriter->endElement(); // office:automatic-styles
    }

    /**
     * Write automatic styles.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeTextStyles(XMLWriter $xmlWriter)
    {
        $styles = Style::getStyles();
        $paragraphStyleCount = 0;

        $style = new Paragraph();
        $style->setStyleName('PB');
        $style->setAuto();
        $styleWriter = new ParagraphStyleWriter($xmlWriter, $style);
        $styleWriter->write();

        $sects = $this->getParentWriter()->getPhpWord()->getSections();
        $countsects = count($sects);
        for ($i = 0; $i < $countsects; ++$i) {
            $iplus1 = $i + 1;
            $style = new Paragraph();
            $style->setStyleName("SB$iplus1");
            $style->setAuto();
            $pnstart = $sects[$i]->getStyle()->getPageNumberingStart();
            $style->setNumLevel($pnstart);
            $styleWriter = new ParagraphStyleWriter($xmlWriter, $style);
            $styleWriter->write();
        }

        foreach ($styles as $style) {
            $sty = $style->getStyleName();
            if (substr($sty, 0, 8) === 'Heading_') {
                $style = new Paragraph();
                $style->setStyleName('HD' . substr($sty, 8));
                $style->setAuto();
                $styleWriter = new ParagraphStyleWriter($xmlWriter, $style);
                $styleWriter->write();
                $style = new Paragraph();
                $style->setStyleName('HE' . substr($sty, 8));
                $style->setAuto();
                $styleWriter = new ParagraphStyleWriter($xmlWriter, $style);
                $styleWriter->write();
            }
        }

        foreach ($styles as $style) {
            if ($style->isAuto() === true) {
                $styleClass = str_replace('\\Style\\', '\\Writer\\ODText\\Style\\', get_class($style));
                if (class_exists($styleClass)) {
                    /** @var \PhpOffice\PhpWord\Writer\ODText\Style\AbstractStyle $styleWriter Type hint */
                    $styleWriter = new $styleClass($xmlWriter, $style);
                    $styleWriter->write();
                }
                if ($style instanceof Paragraph) {
                    $paragraphStyleCount++;
                }
            }
        }
        foreach ($this->imageParagraphStyles as $style) {
            $styleWriter = new \PhpOffice\PhpWord\Writer\ODText\Style\Paragraph($xmlWriter, $style);
            $styleWriter->write();
        }
    }

    /**
     * Get automatic styles.
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    private function getAutoStyles(PhpWord $phpWord)
    {
        $sections = $phpWord->getSections();
        $paragraphStyleCount = 0;
        $fontStyleCount = 0;
        foreach ($sections as $section) {
            $style = $section->getStyle();
            $style->setStyleName("Section{$section->getSectionId()}");
            $this->autoStyles['Section'][] = $style;
            $this->getContainerStyle($section, $paragraphStyleCount, $fontStyleCount);
        }
    }

    /**
     * Get all styles of each elements in container recursively
     *
     * Table style can be null or string of the style name
     *
     * @param \PhpOffice\PhpWord\Element\AbstractContainer $container
     * @param int $paragraphStyleCount
     * @param int $fontStyleCount
     * @todo Simplify the logic
     */
    private function getContainerStyle($container, &$paragraphStyleCount, &$fontStyleCount)
    {
        $elements = $container->getElements();
        foreach ($elements as $element) {
            if ($element instanceof TextRun) {
                $this->getElementStyleTextRun($element, $paragraphStyleCount);
                $this->getContainerStyle($element, $paragraphStyleCount, $fontStyleCount);
            } elseif ($element instanceof Text) {
                $this->getElementStyle($element, $paragraphStyleCount, $fontStyleCount);
            } elseif ($element instanceof Field) {
                $this->getElementStyleField($element, $fontStyleCount);
            } elseif ($element instanceof Image) {
                $style = $element->getStyle();
                $style->setStyleName('fr' . $element->getMediaIndex());
                $this->autoStyles['Image'][] = $style;
                $sty = new \PhpOffice\PhpWord\Style\Paragraph();
                $sty->setStyleName('IM' . $element->getMediaIndex());
                $sty->setAuto();
                $sty->setAlignment($style->getAlignment());
                $this->imageParagraphStyles[] = $sty;
            } elseif ($element instanceof Table) {
                /** @var \PhpOffice\PhpWord\Style\Table $style */
                $style = $element->getStyle();
                if (is_string($style)) {
                    $style = Style::getStyle($style);
                }
                if ($style === null) {
                    $style = new TableStyle();
                }
                $style->setStyleName($element->getElementId());
                $style->setColumnWidths($element->findFirstDefinedCellWidths());
                $this->autoStyles['Table'][] = $style;
            }
        }
    }

    /**
     * Get style of individual element
     *
     * @param \PhpOffice\PhpWord\Element\Text $element
     * @param int $paragraphStyleCount
     * @param int $fontStyleCount
     */
    private function getElementStyle($element, &$paragraphStyleCount, &$fontStyleCount)
    {
        $fontStyle = $element->getFontStyle();
        $paragraphStyle = $element->getParagraphStyle();
        $phpWord = $this->getParentWriter()->getPhpWord();

        if ($fontStyle instanceof Font) {
            // Font
            $name = $fontStyle->getStyleName();
            if (!$name) {
                $fontStyleCount++;
                $style = $phpWord->addFontStyle("T{$fontStyleCount}", $fontStyle, null);
                $style->setAuto();
                $style->setParagraph(null);
                $element->setFontStyle("T{$fontStyleCount}");
            } else {
                $element->setFontStyle($name);
            }
        }
        if ($paragraphStyle instanceof Paragraph) {
            // Paragraph
            $name = $paragraphStyle->getStyleName();
            if (!$name) {
                $paragraphStyleCount++;
                $style = $phpWord->addParagraphStyle("P{$paragraphStyleCount}", $paragraphStyle);
                $style->setAuto();
                $element->setParagraphStyle("P{$paragraphStyleCount}");
            } else {
                $element->setParagraphStyle($name);
            }
        } elseif ($paragraphStyle) {
            $paragraphStyleCount++;
            $parstylename = "P$paragraphStyleCount" . "_$paragraphStyle";
            $style = $phpWord->addParagraphStyle($parstylename, $paragraphStyle);
            $style->setAuto();
            $element->setParagraphStyle($parstylename);
        }
    }

    /**
     * Get font style of individual field element
     *
     * @param \PhpOffice\PhpWord\Element\Field $element
     * @param int $paragraphStyleCount
     * @param int $fontStyleCount
     */
    private function getElementStyleField($element, &$fontStyleCount)
    {
        $fontStyle = $element->getFontStyle();
        $phpWord = $this->getParentWriter()->getPhpWord();

        if ($fontStyle instanceof Font) {
            $name = $fontStyle->getStyleName();
            if (!$name) {
                $fontStyleCount++;
                $style = $phpWord->addFontStyle("T{$fontStyleCount}", $fontStyle, null);
                $style->setAuto();
                $style->setParagraph(null);
                $element->setFontStyle("T{$fontStyleCount}");
            } else {
                $element->setFontStyle($name);
            }
        }
    }

    /**
     * Get style of individual element
     *
     * @param \PhpOffice\PhpWord\Element\TextRun $element
     * @param int $paragraphStyleCount
     */
    private function getElementStyleTextRun($element, &$paragraphStyleCount)
    {
        $paragraphStyle = $element->getParagraphStyle();
        $phpWord = $this->getParentWriter()->getPhpWord();

        if ($paragraphStyle instanceof Paragraph) {
            // Paragraph
            $name = $paragraphStyle->getStyleName();
            if (!$name) {
                $paragraphStyleCount++;
                $style = $phpWord->addParagraphStyle("P{$paragraphStyleCount}", $paragraphStyle);
                $style->setAuto();
                $element->setParagraphStyle("P{$paragraphStyleCount}");
            } else {
                $element->setParagraphStyle($name);
            }
        } elseif ($paragraphStyle) {
            $paragraphStyleCount++;
            $parstylename = "P$paragraphStyleCount" . "_$paragraphStyle";
            $style = $phpWord->addParagraphStyle($parstylename, $paragraphStyle);
            $style->setAuto();
            $element->setParagraphStyle($parstylename);
        }
    }

    /**
     * Finds all tracked changes
     *
     * @param AbstractContainer $container
     * @param \PhpOffice\PhpWord\Element\AbstractElement[] $trackedChanges
     */
    private function collectTrackedChanges(AbstractContainer $container, &$trackedChanges = array())
    {
        $elements = $container->getElements();
        foreach ($elements as $element) {
            if ($element->getTrackChange() != null) {
                $trackedChanges[] = $element;
            }
            if (is_callable(array($element, 'getElements'))) {
                $this->collectTrackedChanges($element, $trackedChanges);
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/Manifest.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

use PhpOffice\PhpWord\Media;

/**
 * ODText manifest part writer: META-INF/manifest.xml
 */
class Manifest extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $parts = array('content.xml', 'meta.xml', 'styles.xml');
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8');
        $xmlWriter->startElement('manifest:manifest');
        $xmlWriter->writeAttribute('manifest:version', '1.2');
        $xmlWriter->writeAttribute('xmlns:manifest', 'urn:oasis:names:tc:opendocument:xmlns:manifest:1.0');

        $xmlWriter->startElement('manifest:file-entry');
        $xmlWriter->writeAttribute('manifest:media-type', 'application/vnd.oasis.opendocument.text');
        $xmlWriter->writeAttribute('manifest:full-path', '/');
        $xmlWriter->writeAttribute('manifest:version', '1.2');
        $xmlWriter->endElement();

        // Parts
        foreach ($parts as $part) {
            $xmlWriter->startElement('manifest:file-entry');
            $xmlWriter->writeAttribute('manifest:media-type', 'text/xml');
            $xmlWriter->writeAttribute('manifest:full-path', $part);
            $xmlWriter->endElement();
        }

        // Media files
        $media = Media::getElements('section');
        foreach ($media as $medium) {
            if ($medium['type'] == 'image') {
                $xmlWriter->startElement('manifest:file-entry');
                $xmlWriter->writeAttribute('manifest:media-type', $medium['imageType']);
                $xmlWriter->writeAttribute('manifest:full-path', 'Pictures/' . $medium['target']);
                $xmlWriter->endElement();
            }
        }

        $xmlWriter->endElement(); // manifest:manifest

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/Meta.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * ODText meta part writer: meta.xml
 *
 * @since 0.11.0
 */
class Meta extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $docProps = $phpWord->getDocInfo();
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8');
        $xmlWriter->startElement('office:document-meta');
        $xmlWriter->writeAttribute('office:version', '1.2');
        $xmlWriter->writeAttribute('xmlns:office', 'urn:oasis:names:tc:opendocument:xmlns:office:1.0');
        $xmlWriter->writeAttribute('xmlns:xlink', 'http://www.w3.org/1999/xlink');
        $xmlWriter->writeAttribute('xmlns:dc', 'http://purl.org/dc/elements/1.1/');
        $xmlWriter->writeAttribute('xmlns:meta', 'urn:oasis:names:tc:opendocument:xmlns:meta:1.0');
        $xmlWriter->writeAttribute('xmlns:ooo', 'http://openoffice.org/2004/office');
        $xmlWriter->writeAttribute('xmlns:grddl', 'http://www.w3.org/2003/g/data-view#');
        $xmlWriter->startElement('office:meta');

        // Core properties
        $xmlWriter->writeElement('dc:title', $docProps->getTitle());
        $xmlWriter->writeElement('dc:subject', $docProps->getSubject());
        $xmlWriter->writeElement('dc:description', $docProps->getDescription());
        $xmlWriter->writeElement('dc:creator', $docProps->getLastModifiedBy());
        $xmlWriter->writeElement('dc:date', gmdate($this->dateFormat, $docProps->getModified()));

        // Extended properties
        $xmlWriter->writeElement('meta:generator', 'PHPWord');
        $xmlWriter->writeElement('meta:initial-creator', $docProps->getCreator());
        $xmlWriter->writeElement('meta:creation-date', gmdate($this->dateFormat, $docProps->getCreated()));
        $xmlWriter->writeElement('meta:keyword', $docProps->getKeywords());

        // Category, company, and manager are put in meta namespace
        $properties = array('Category', 'Company', 'Manager');
        foreach ($properties as $property) {
            $method = "get{$property}";
            if ($docProps->$method() !== null) {
                $this->writeCustomProperty($xmlWriter, $property, $docProps->$method());
            }
        }

        // Other custom properties
        // @todo Check type. Currently all assumed as string
        foreach ($docProps->getCustomProperties() as $property) {
            $value = $docProps->getCustomPropertyValue($property);
            $this->writeCustomProperty($xmlWriter, $property, $value);
        }

        $xmlWriter->endElement(); // office:meta
        $xmlWriter->endElement(); // office:document-meta

        return $xmlWriter->getData();
    }

    /**
     * Write individual property
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $property
     * @param string $value
     *
     * @todo Handle other `$type`: double|date|dateTime|duration|boolean (4th arguments)
     */
    private function writeCustomProperty(XMLWriter $xmlWriter, $property, $value)
    {
        $xmlWriter->startElement('meta:user-defined');
        $xmlWriter->writeAttribute('meta:name', $property);
        // if ($type !== null) {
        //     $xmlWriter->writeAttribute('meta:value-type', $type);
        // }
        $this->writeText($value);
        $xmlWriter->endElement(); // meta:user-defined
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/Mimetype.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

/**
 * ODText mimetype part writer: mimetype
 */
class Mimetype extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        return 'application/vnd.oasis.opendocument.text';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Part/Styles.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Part;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\Converter;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;

/**
 * ODText styles part writer: styles.xml
 */
class Styles extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        // XML header
        $xmlWriter->startDocument('1.0', 'UTF-8');
        $xmlWriter->startElement('office:document-styles');
        $this->writeCommonRootAttributes($xmlWriter);

        // Font declarations
        $this->writeFontFaces($xmlWriter);

        // Office styles
        $xmlWriter->startElement('office:styles');
        $this->writeDefault($xmlWriter);
        $this->writeNamed($xmlWriter);
        $xmlWriter->endElement();

        // Automatic styles
        $xmlWriter->startElement('office:automatic-styles');
        $this->writePageLayout($xmlWriter);
        $xmlWriter->endElement(); // office:automatic-styles

        // Master style
        $this->writeMaster($xmlWriter);

        $xmlWriter->endElement(); // office:document-styles

        return $xmlWriter->getData();
    }

    /**
     * Write default styles.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeDefault(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('style:default-style');
        $xmlWriter->writeAttribute('style:family', 'paragraph');

        // Paragraph
        $xmlWriter->startElement('style:paragraph-properties');
        $xmlWriter->writeAttribute('fo:hyphenation-ladder-count', 'no-limit');
        $xmlWriter->writeAttribute('style:text-autospace', 'ideograph-alpha');
        $xmlWriter->writeAttribute('style:punctuation-wrap', 'hanging');
        $xmlWriter->writeAttribute('style:line-break', 'strict');
        $xmlWriter->writeAttribute('style:tab-stop-distance', '1.249cm');
        $xmlWriter->writeAttribute('style:writing-mode', 'page');
        $xmlWriter->endElement(); // style:paragraph-properties

        $language = $this->getParentWriter()->getPhpWord()->getSettings()->getThemeFontLang();
        $latinLang = $language != null && is_string($language->getLatin()) ? explode('-', $language->getLatin()) : array('fr', 'FR');
        $asianLang = $language != null && is_string($language->getEastAsia()) ? explode('-', $language->getEastAsia()) : array('zh', 'CN');
        $complexLang = $language != null && is_string($language->getBidirectional()) ? explode('-', $language->getBidirectional()) : array('hi', 'IN');
        if ($this->getParentWriter()->getPhpWord()->getSettings()->hasHideGrammaticalErrors()) {
            $latinLang = $asianLang = $complexLang = array('zxx', 'none');
        }

        // Font
        $xmlWriter->startElement('style:text-properties');
        $xmlWriter->writeAttribute('style:use-window-font-color', 'true');
        $xmlWriter->writeAttribute('style:font-name', Settings::getDefaultFontName());
        $xmlWriter->writeAttribute('fo:font-size', Settings::getDefaultFontSize() . 'pt');
        $xmlWriter->writeAttribute('fo:language', $latinLang[0]);
        $xmlWriter->writeAttribute('fo:country', $latinLang[1]);
        $xmlWriter->writeAttribute('style:letter-kerning', 'true');
        $xmlWriter->writeAttribute('style:font-name-asian', Settings::getDefaultFontName() . '2');
        $xmlWriter->writeAttribute('style:font-size-asian', Settings::getDefaultFontSize() . 'pt');
        $xmlWriter->writeAttribute('style:language-asian', $asianLang[0]);
        $xmlWriter->writeAttribute('style:country-asian', $asianLang[1]);
        $xmlWriter->writeAttribute('style:font-name-complex', Settings::getDefaultFontName() . '2');
        $xmlWriter->writeAttribute('style:font-size-complex', Settings::getDefaultFontSize() . 'pt');
        $xmlWriter->writeAttribute('style:language-complex', $complexLang[0]);
        $xmlWriter->writeAttribute('style:country-complex', $complexLang[1]);
        $xmlWriter->writeAttribute('fo:hyphenate', 'false');
        $xmlWriter->writeAttribute('fo:hyphenation-remain-char-count', '2');
        $xmlWriter->writeAttribute('fo:hyphenation-push-char-count', '2');
        $xmlWriter->endElement(); // style:text-properties

        $xmlWriter->endElement(); // style:default-style
    }

    /**
     * Write named styles.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeNamed(XMLWriter $xmlWriter)
    {
        $styles = Style::getStyles();
        if (count($styles) > 0) {
            foreach ($styles as $style) {
                if ($style->isAuto() === false) {
                    $styleClass = str_replace('\\Style\\', '\\Writer\\ODText\\Style\\', get_class($style));
                    if (class_exists($styleClass)) {
                        /** @var $styleWriter \PhpOffice\PhpWord\Writer\ODText\Style\AbstractStyle Type hint */
                        $styleWriter = new $styleClass($xmlWriter, $style);
                        $styleWriter->write();
                    }
                }
            }
        }
    }

    /**
     * Convert int in twips to inches/cm then to string and append unit
     *
     * @param int|float $twips
     * @param float $factor
     * return string
     */
    private static function cvttwiptostr($twips, $factor = 1.0)
    {
        $ins = (string) ($twips * $factor / Converter::INCH_TO_TWIP) . 'in';
        $cms = (string) ($twips * $factor * Converter::INCH_TO_CM / Converter::INCH_TO_TWIP) . 'cm';

        return (strlen($ins) < strlen($cms)) ? $ins : $cms;
    }

    /**
     * call writePageLayoutIndiv to write page layout styles for each page
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writePageLayout(XMLWriter $xmlWriter)
    {
        $sections = $this->getParentWriter()->getPhpWord()->getSections();
        $countsects = count($sections);
        for ($i = 0; $i < $countsects; ++$i) {
            $this->writePageLayoutIndiv($xmlWriter, $sections[$i], $i + 1);
        }
    }

    /**
     * Write page layout styles.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Section $section
     * @param int $sectionNbr
     */
    private function writePageLayoutIndiv(XMLWriter $xmlWriter, $section, $sectionNbr)
    {
        $sty = $section->getStyle();
        if (count($section->getHeaders()) > 0) {
            $topfactor = 0.5;
        } else {
            $topfactor = 1.0;
        }
        if (count($section->getFooters()) > 0) {
            $botfactor = 0.5;
        } else {
            $botfactor = 1.0;
        }
        $orient = $sty->getOrientation();
        $pwidth = self::cvttwiptostr($sty->getPageSizeW());
        $pheight = self::cvttwiptostr($sty->getPageSizeH());
        $mtop = self::cvttwiptostr($sty->getMarginTop(), $topfactor);
        $mbottom = self::cvttwiptostr($sty->getMarginBottom(), $botfactor);
        $mleft = self::cvttwiptostr($sty->getMarginRight());
        $mright = self::cvttwiptostr($sty->getMarginLeft());

        $xmlWriter->startElement('style:page-layout');
        $xmlWriter->writeAttribute('style:name', "Mpm$sectionNbr");

        $xmlWriter->startElement('style:page-layout-properties');
        $xmlWriter->writeAttribute('fo:page-width', $pwidth);
        $xmlWriter->writeAttribute('fo:page-height', $pheight);
        $xmlWriter->writeAttribute('style:num-format', '1');
        $xmlWriter->writeAttribute('style:print-orientation', $orient);
        $xmlWriter->writeAttribute('fo:margin-top', $mtop);
        $xmlWriter->writeAttribute('fo:margin-bottom', $mbottom);
        $xmlWriter->writeAttribute('fo:margin-left', $mleft);
        $xmlWriter->writeAttribute('fo:margin-right', $mright);
        $xmlWriter->writeAttribute('style:writing-mode', 'lr-tb');
        $xmlWriter->writeAttribute('style:layout-grid-color', '#c0c0c0');
        $xmlWriter->writeAttribute('style:layout-grid-lines', '25199');
        $xmlWriter->writeAttribute('style:layout-grid-base-height', '0.423cm');
        $xmlWriter->writeAttribute('style:layout-grid-ruby-height', '0cm');
        $xmlWriter->writeAttribute('style:layout-grid-mode', 'none');
        $xmlWriter->writeAttribute('style:layout-grid-ruby-below', 'false');
        $xmlWriter->writeAttribute('style:layout-grid-print', 'false');
        $xmlWriter->writeAttribute('style:layout-grid-display', 'false');
        $xmlWriter->writeAttribute('style:layout-grid-base-width', '0.37cm');
        $xmlWriter->writeAttribute('style:layout-grid-snap-to', 'true');
        $xmlWriter->writeAttribute('style:footnote-max-height', '0cm');

        $xmlWriter->startElement('style:footnote-sep');
        $xmlWriter->writeAttribute('style:width', '0.018cm');
        $xmlWriter->writeAttribute('style:line-style', 'solid');
        $xmlWriter->writeAttribute('style:adjustment', 'left');
        $xmlWriter->writeAttribute('style:rel-width', '25%');
        $xmlWriter->writeAttribute('style:color', '#000000');
        $xmlWriter->endElement(); //style:footnote-sep

        $xmlWriter->endElement(); // style:page-layout-properties

        $xmlWriter->startElement('style:header-style');
        if ($topfactor < 1.0) {
            $xmlWriter->startElement('style:header-footer-properties');
            $xmlWriter->writeAttribute('fo:min-height', $mtop);
            $xmlWriter->writeAttribute('fo:margin-bottom', $mtop);
            $xmlWriter->writeAttribute('style:dynamic-spacing', 'true');
            $xmlWriter->endElement(); // style:header-footer-properties
        }
        $xmlWriter->endElement(); // style:header-style

        $xmlWriter->startElement('style:footer-style');
        if ($botfactor < 1.0) {
            $xmlWriter->startElement('style:header-footer-properties');
            $xmlWriter->writeAttribute('fo:min-height', $mbottom);
            $xmlWriter->writeAttribute('fo:margin-top', $mbottom);
            $xmlWriter->writeAttribute('style:dynamic-spacing', 'true');
            $xmlWriter->endElement(); // style:header-footer-properties
        }
        $xmlWriter->endElement(); // style:footer-style

        $xmlWriter->endElement(); // style:page-layout
    }

    /**
     * Write master style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeMaster(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('office:master-styles');

        $sections = $this->getParentWriter()->getPhpWord()->getSections();
        $countsects = count($sections);
        for ($i = 0; $i < $countsects; ++$i) {
            $iplus1 = $i + 1;
            $xmlWriter->startElement('style:master-page');
            $xmlWriter->writeAttribute('style:name', "Standard$iplus1");
            $xmlWriter->writeAttribute('style:page-layout-name', "Mpm$iplus1");
            // Multiple headers and footers probably not supported,
            //   and, even if they are, I'm not sure how,
            //   so quit after generating one.
            foreach ($sections[$i]->getHeaders() as $hdr) {
                $xmlWriter->startElement('style:header');
                foreach ($hdr->getElements() as $elem) {
                    $cl1 = get_class($elem);
                    $cl2 = str_replace('\\Element\\', '\\Writer\\ODText\\Element\\', $cl1);
                    if (class_exists($cl2)) {
                        $wtr = new $cl2($xmlWriter, $elem);
                        $wtr->write();
                    }
                }
                $xmlWriter->endElement(); // style:header
                break;
            }
            foreach ($sections[$i]->getFooters() as $hdr) {
                $xmlWriter->startElement('style:footer');
                foreach ($hdr->getElements() as $elem) {
                    $cl1 = get_class($elem);
                    $cl2 = str_replace('\\Element\\', '\\Writer\\ODText\\Element\\', $cl1);
                    if (class_exists($cl2)) {
                        $wtr = new $cl2($xmlWriter, $elem);
                        $wtr->write();
                    }
                }
                $xmlWriter->endElement(); // style:footer
                break;
            }
            $xmlWriter->endElement(); // style:master-page
        }
        $xmlWriter->endElement(); // office:master-styles
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/AbstractStyle.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

use PhpOffice\PhpWord\Writer\Word2007\Style\AbstractStyle as Word2007AbstractStyle;

/**
 * Style writer
 *
 * @since 0.10.0
 */
abstract class AbstractStyle extends Word2007AbstractStyle
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/Font.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

/**
 * Font style writer
 *
 * @since 0.10.0
 */
class Font extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Font) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $stylep = (method_exists($style, 'getParagraph')) ? $style->getParagraph() : null;
        if ($stylep instanceof \PhpOffice\PhpWord\Style\Paragraph) {
            $temp1 = clone $stylep;
            $temp1->setStyleName($style->getStyleName());
            $temp2 = new \PhpOffice\PhpWord\Writer\ODText\Style\Paragraph($xmlWriter, $temp1);
            $temp2->write();
        }

        $xmlWriter->startElement('style:style');
        $xmlWriter->writeAttribute('style:name', $style->getStyleName());
        $xmlWriter->writeAttribute('style:family', 'text');
        $xmlWriter->startElement('style:text-properties');

        // Name
        $font = $style->getName();
        $xmlWriter->writeAttributeIf($font != '', 'style:font-name', $font);
        $xmlWriter->writeAttributeIf($font != '', 'style:font-name-complex', $font);
        $size = $style->getSize();

        // Size
        $xmlWriter->writeAttributeIf(is_numeric($size), 'fo:font-size', $size . 'pt');
        $xmlWriter->writeAttributeIf(is_numeric($size), 'style:font-size-asian', $size . 'pt');
        $xmlWriter->writeAttributeIf(is_numeric($size), 'style:font-size-complex', $size . 'pt');

        // Color
        $color = $style->getColor();
        $xmlWriter->writeAttributeIf($color != '', 'fo:color', '#' . \PhpOffice\PhpWord\Shared\Converter::stringToRgb($color));

        // Bold & italic
        $xmlWriter->writeAttributeIf($style->isBold(), 'fo:font-weight', 'bold');
        $xmlWriter->writeAttributeIf($style->isBold(), 'style:font-weight-asian', 'bold');
        $xmlWriter->writeAttributeIf($style->isItalic(), 'fo:font-style', 'italic');
        $xmlWriter->writeAttributeIf($style->isItalic(), 'style:font-style-asian', 'italic');
        $xmlWriter->writeAttributeIf($style->isItalic(), 'style:font-style-complex', 'italic');

        // Underline
        // @todo Various mode of underline
        $underline = $style->getUnderline();
        $xmlWriter->writeAttributeIf($underline != 'none', 'style:text-underline-style', 'solid');

        // Strikethrough, double strikethrough
        $xmlWriter->writeAttributeIf($style->isStrikethrough(), 'style:text-line-through-type', 'single');
        $xmlWriter->writeAttributeIf($style->isDoubleStrikethrough(), 'style:text-line-through-type', 'double');

        // Small caps, all caps
        $xmlWriter->writeAttributeIf($style->isSmallCaps(), 'fo:font-variant', 'small-caps');
        $xmlWriter->writeAttributeIf($style->isAllCaps(), 'fo:text-transform', 'uppercase');

        //Hidden text
        $xmlWriter->writeAttributeIf($style->isHidden(), 'text:display', 'none');

        // Superscript/subscript
        $xmlWriter->writeAttributeIf($style->isSuperScript(), 'style:text-position', 'super');
        $xmlWriter->writeAttributeIf($style->isSubScript(), 'style:text-position', 'sub');

        if ($style->isNoProof()) {
            $xmlWriter->writeAttribute('fo:language', 'zxx');
            $xmlWriter->writeAttribute('style:language-asian', 'zxx');
            $xmlWriter->writeAttribute('style:language-complex', 'zxx');
            $xmlWriter->writeAttribute('fo:country', 'none');
            $xmlWriter->writeAttribute('style:country-asian', 'none');
            $xmlWriter->writeAttribute('style:country-complex', 'none');
        }

        // @todo Foreground-Color

        // @todo Background color

        $xmlWriter->endElement(); // style:text-properties
        $xmlWriter->endElement(); // style:style
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

/**
 * Image style writer
 *
 * @since 0.11.0
 */
class Image extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Style\Image $style Type hint */
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Image) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('style:style');
        $xmlWriter->writeAttribute('style:name', $style->getStyleName());
        $xmlWriter->writeAttribute('style:family', 'graphic');
        $xmlWriter->writeAttribute('style:parent-style-name', 'Graphics');
        $xmlWriter->startElement('style:graphic-properties');
        $xmlWriter->writeAttribute('style:vertical-pos', 'top');
        $xmlWriter->writeAttribute('style:vertical-rel', 'baseline');
        $xmlWriter->endElement(); // style:graphic-properties
        $xmlWriter->endElement(); // style:style
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/Paragraph.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

use PhpOffice\PhpWord\Shared\Converter;

/**
 * Font style writer
 *
 * @since 0.10.0
 */
class Paragraph extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Paragraph) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $marginTop = $style->getSpaceBefore();
        $marginBottom = $style->getSpaceAfter();

        $xmlWriter->startElement('style:style');

        $styleName = $style->getStyleName();
        $styleAuto = false;
        $mpm = '';
        $psm = '';
        $pagestart = -1;
        $breakafter = $breakbefore = $breakauto = false;
        if ($style->isAuto()) {
            if (substr($styleName, 0, 2) === 'PB') {
                $styleAuto = true;
                $breakafter = true;
            } elseif (substr($styleName, 0, 2) === 'SB') {
                $styleAuto = true;
                $mpm = 'Standard' . substr($styleName, 2);
                $psn = $style->getNumLevel();
                $pagestart = $psn;
            } elseif (substr($styleName, 0, 2) === 'HD') {
                $styleAuto = true;
                $psm = 'Heading_' . substr($styleName, 2);
                $stylep = \PhpOffice\PhpWord\Style::getStyle($psm);
                if ($stylep instanceof \PhpOffice\PhpWord\Style\Font) {
                    if (method_exists($stylep, 'getParagraph')) {
                        $stylep = $stylep->getParagraph();
                    }
                }
                if ($stylep instanceof \PhpOffice\PhpWord\Style\Paragraph) {
                    if ($stylep->hasPageBreakBefore()) {
                        $breakbefore = true;
                    }
                }
            } elseif (substr($styleName, 0, 2) === 'HE') {
                $styleAuto = true;
                $psm = 'Heading_' . substr($styleName, 2);
                $breakauto = true;
            } else {
                $styleAuto = true;
                $psm = 'Normal';
                if (preg_match('/^P\\d+_(\\w+)$/', $styleName, $matches)) {
                    $psm = $matches[1];
                }
            }
        }

        $xmlWriter->writeAttribute('style:name', $style->getStyleName());
        $xmlWriter->writeAttribute('style:family', 'paragraph');
        if ($styleAuto) {
            $xmlWriter->writeAttributeIf($psm !== '', 'style:parent-style-name', $psm);
            $xmlWriter->writeAttributeIf($mpm !== '', 'style:master-page-name', $mpm);
        }

        $xmlWriter->startElement('style:paragraph-properties');
        if ($styleAuto) {
            if ($breakafter) {
                $xmlWriter->writeAttribute('fo:break-after', 'page');
                $xmlWriter->writeAttribute('fo:margin-top', '0cm');
                $xmlWriter->writeAttribute('fo:margin-bottom', '0cm');
            } elseif ($breakbefore) {
                $xmlWriter->writeAttribute('fo:break-before', 'page');
            } elseif ($breakauto) {
                $xmlWriter->writeAttribute('fo:break-before', 'auto');
            }
            if ($pagestart > 0) {
                $xmlWriter->writeAttribute('style:page-number', $pagestart);
            }
        }
        if (!$breakafter && !$breakbefore && !$breakauto) {
            $twipToPoint = Converter::INCH_TO_TWIP / Converter::INCH_TO_POINT; // 20
            $xmlWriter->writeAttributeIf($marginTop !== null, 'fo:margin-top', ($marginTop / $twipToPoint) . 'pt');
            $xmlWriter->writeAttributeIf($marginBottom !== null, 'fo:margin-bottom', ($marginBottom / $twipToPoint) . 'pt');
        }
        $temp = $style->getAlignment();
        $xmlWriter->writeAttributeIf($temp !== '', 'fo:text-align', $temp);
        $temp = $style->getLineHeight();
        $xmlWriter->writeAttributeIf($temp !== null, 'fo:line-height', ((string) ($temp * 100) . '%'));
        $xmlWriter->writeAttributeIf($style->hasPageBreakBefore() === true, 'fo:break-before', 'page');

        $tabs = $style->getTabs();
        if ($tabs !== null && count($tabs) > 0) {
            $xmlWriter->startElement('style:tab-stops');
            foreach ($tabs as $tab) {
                $xmlWriter->startElement('style:tab-stop');
                $xmlWriter->writeAttribute('style:type', $tab->getType());
                $xmlWriter->writeAttribute('style:position', (string) ($tab->getPosition() / Converter::INCH_TO_TWIP) . 'in');
                $xmlWriter->endElement();
            }
            $xmlWriter->endElement();
        }

        //Right to left
        $xmlWriter->writeAttributeIf($style->isBidi(), 'style:writing-mode', 'rl-tb');

        //Indentation
        $indent = $style->getIndentation();
        //if ($indent instanceof \PhpOffice\PhpWord\Style\Indentation) {
        if (!empty($indent)) {
            $marg = $indent->getLeft();
            $xmlWriter->writeAttributeIf($marg !== null, 'fo:margin-left', (string) ($marg / Converter::INCH_TO_TWIP) . 'in');
            $marg = $indent->getRight();
            $xmlWriter->writeAttributeIf($marg !== null, 'fo:margin-right', (string) ($marg / Converter::INCH_TO_TWIP) . 'in');
        }

        $xmlWriter->endElement(); //style:paragraph-properties

        if ($styleAuto && substr($styleName, 0, 2) === 'SB') {
            $xmlWriter->startElement('style:text-properties');
            $xmlWriter->writeAttribute('text:display', 'none');
            $xmlWriter->endElement();
        }

        $xmlWriter->endElement(); //style:style
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/Section.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

/**
 * Section style writer
 *
 * @since 0.11.0
 */
class Section extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Style\Section $style Type hint */
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Section) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('style:style');
        $xmlWriter->writeAttribute('style:name', $style->getStyleName());
        $xmlWriter->writeAttribute('style:family', 'section');
        $xmlWriter->startElement('style:section-properties');

        $xmlWriter->startElement('style:columns');
        $xmlWriter->writeAttribute('fo:column-count', $style->getColsNum());
        $xmlWriter->endElement(); // style:columns

        $xmlWriter->endElement(); // style:section-properties
        $xmlWriter->endElement(); // style:style
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText/Style/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\ODText\Style;

/**
 * Table style writer
 *
 * @since 0.11.0
 */
class Table extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Style\Table $style Type hint */
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Table) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('style:style');
        $xmlWriter->writeAttribute('style:name', $style->getStyleName());
        $xmlWriter->writeAttribute('style:family', 'table');
        $xmlWriter->startElement('style:table-properties');
        //$xmlWriter->writeAttribute('style:width', 'table');
        $xmlWriter->writeAttribute('style:rel-width', 100);
        $xmlWriter->writeAttribute('table:align', 'center');
        $xmlWriter->writeAttributeIf($style->isBidiVisual(), 'style:writing-mode', 'rl-tb');
        $xmlWriter->endElement(); // style:table-properties
        $xmlWriter->endElement(); // style:style

        $cellWidths = $style->getColumnWidths();
        $countCellWidths = $cellWidths === null ? 0 : count($cellWidths);

        for ($i = 0; $i < $countCellWidths; $i++) {
            $width = $cellWidths[$i];
            $xmlWriter->startElement('style:style');
            $xmlWriter->writeAttribute('style:name', $style->getStyleName() . '.' . $i);
            $xmlWriter->writeAttribute('style:family', 'table-column');
            $xmlWriter->startElement('style:table-column-properties');
            $xmlWriter->writeAttribute('style:column-width', number_format($width * 0.0017638889, 2, '.', '') . 'cm');
            $xmlWriter->endElement(); // style:table-column-properties
            $xmlWriter->endElement(); // style:style
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/ODText.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\Media;
use PhpOffice\PhpWord\PhpWord;

/**
 * ODText writer
 *
 * @since 0.7.0
 */
class ODText extends AbstractWriter implements WriterInterface
{
    /**
     * Create new ODText writer
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function __construct(PhpWord $phpWord = null)
    {
        // Assign PhpWord
        $this->setPhpWord($phpWord);

        // Create parts
        $this->parts = array(
            'Mimetype'  => 'mimetype',
            'Content'   => 'content.xml',
            'Meta'      => 'meta.xml',
            'Styles'    => 'styles.xml',
            'Manifest'  => 'META-INF/manifest.xml',
        );
        foreach (array_keys($this->parts) as $partName) {
            $partClass = get_class($this) . '\\Part\\' . $partName;
            if (class_exists($partClass)) {
                /** @var $partObject \PhpOffice\PhpWord\Writer\ODText\Part\AbstractPart Type hint */
                $partObject = new $partClass();
                $partObject->setParentWriter($this);
                $this->writerParts[strtolower($partName)] = $partObject;
            }
        }

        // Set package paths
        $this->mediaPaths = array('image' => 'Pictures/');
    }

    /**
     * Save PhpWord to file.
     *
     * @param string $filename
     */
    public function save($filename = null)
    {
        $filename = $this->getTempFile($filename);
        $zip = $this->getZipArchive($filename);

        // Add section media files
        $sectionMedia = Media::getElements('section');
        if (!empty($sectionMedia)) {
            $this->addFilesToPackage($zip, $sectionMedia);
        }

        // Write parts
        foreach ($this->parts as $partName => $fileName) {
            if ($fileName != '') {
                $zip->addFromString($fileName, $this->getWriterPart($partName)->write());
            }
        }

        // Close zip archive and cleanup temp file
        $zip->close();
        $this->cleanupTempFile();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/PDF/AbstractRenderer.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PhpWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\PDF;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Writer\HTML;

/**
 * Abstract PDF renderer
 *
 * @since 0.10.0
 */
abstract class AbstractRenderer extends HTML
{
    /**
     * Name of renderer include file
     *
     * @var string
     */
    protected $includeFile;

    /**
     * Temporary storage directory
     *
     * @var string
     */
    protected $tempDir = '';

    /**
     * Font
     *
     * @var string
     */
    protected $font;

    /**
     * Paper size
     *
     * @var int
     */
    protected $paperSize;

    /**
     * Orientation
     *
     * @var string
     */
    protected $orientation;

    /**
     * Paper Sizes xRef List
     *
     * @var array
     */
    protected static $paperSizes = array(
        9 => 'A4', // (210 mm by 297 mm)
    );

    /**
     * Create new instance
     *
     * @param PhpWord $phpWord PhpWord object
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function __construct(PhpWord $phpWord)
    {
        parent::__construct($phpWord);
        if ($this->includeFile != null) {
            $includeFile = Settings::getPdfRendererPath() . '/' . $this->includeFile;
            if (file_exists($includeFile)) {
                /** @noinspection PhpIncludeInspection Dynamic includes */
                require_once $includeFile;
            } else {
                // @codeCoverageIgnoreStart
                // Can't find any test case. Uncomment when found.
                throw new Exception('Unable to load PDF Rendering library');
                // @codeCoverageIgnoreEnd
            }
        }
    }

    /**
     * Get Font
     *
     * @return string
     */
    public function getFont()
    {
        return $this->font;
    }

    /**
     * Set font. Examples:
     *      'arialunicid0-chinese-simplified'
     *      'arialunicid0-chinese-traditional'
     *      'arialunicid0-korean'
     *      'arialunicid0-japanese'
     *
     * @param string $fontName
     * @return self
     */
    public function setFont($fontName)
    {
        $this->font = $fontName;

        return $this;
    }

    /**
     * Get Paper Size
     *
     * @return int
     */
    public function getPaperSize()
    {
        return $this->paperSize;
    }

    /**
     * Set Paper Size
     *
     * @param int $value Paper size = PAPERSIZE_A4
     * @return self
     */
    public function setPaperSize($value = 9)
    {
        $this->paperSize = $value;

        return $this;
    }

    /**
     * Get Orientation
     *
     * @return string
     */
    public function getOrientation()
    {
        return $this->orientation;
    }

    /**
     * Set Orientation
     *
     * @param string $value Page orientation ORIENTATION_DEFAULT
     * @return self
     */
    public function setOrientation($value = 'default')
    {
        $this->orientation = $value;

        return $this;
    }

    /**
     * Save PhpWord to PDF file, pre-save
     *
     * @param string $filename Name of the file to save as
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     * @return resource
     */
    protected function prepareForSave($filename = null)
    {
        $fileHandle = fopen($filename, 'w');
        // @codeCoverageIgnoreStart
        // Can't find any test case. Uncomment when found.
        if ($fileHandle === false) {
            throw new Exception("Could not open file $filename for writing.");
        }
        // @codeCoverageIgnoreEnd
        $this->isPdf = true;

        return $fileHandle;
    }

    /**
     * Save PhpWord to PDF file, post-save
     *
     * @param resource $fileHandle
     *
     * @throws Exception
     */
    protected function restoreStateAfterSave($fileHandle)
    {
        fclose($fileHandle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/PDF/DomPDF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PhpWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\PDF;

use Dompdf\Dompdf as DompdfLib;
use PhpOffice\PhpWord\Writer\WriterInterface;

/**
 * DomPDF writer
 *
 * @see  https://github.com/dompdf/dompdf
 * @since 0.10.0
 */
class DomPDF extends AbstractRenderer implements WriterInterface
{
    /**
     * Name of renderer include file
     *
     * @var string
     */
    protected $includeFile = null;

    /**
     * Gets the implementation of external PDF library that should be used.
     *
     * @return Dompdf implementation
     */
    protected function createExternalWriterInstance()
    {
        return new DompdfLib();
    }

    /**
     * Save PhpWord to file.
     *
     * @param string $filename Name of the file to save as
     */
    public function save($filename = null)
    {
        $fileHandle = parent::prepareForSave($filename);

        //  PDF settings
        $paperSize = 'A4';
        $orientation = 'portrait';

        //  Create PDF
        $pdf = $this->createExternalWriterInstance();
        $pdf->setPaper(strtolower($paperSize), $orientation);
        $pdf->loadHtml(str_replace(PHP_EOL, '', $this->getContent()));
        $pdf->render();

        //  Write to file
        fwrite($fileHandle, $pdf->output());

        parent::restoreStateAfterSave($fileHandle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/PDF/MPDF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PhpWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\PDF;

use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Writer\WriterInterface;

/**
 * MPDF writer
 *
 * @see  http://www.mpdf1.com/
 * @since 0.11.0
 */
class MPDF extends AbstractRenderer implements WriterInterface
{
    /**
     * Overridden to set the correct includefile, only needed for MPDF 5
     *
     * @codeCoverageIgnore
     * @param PhpWord $phpWord
     */
    public function __construct(PhpWord $phpWord)
    {
        if (file_exists(Settings::getPdfRendererPath() . '/mpdf.php')) {
            // MPDF version 5.* needs this file to be included, later versions not
            $this->includeFile = 'mpdf.php';
        }
        parent::__construct($phpWord);
    }

    /**
     * Gets the implementation of external PDF library that should be used.
     *
     * @return Mpdf implementation
     */
    protected function createExternalWriterInstance()
    {
        $mPdfClass = $this->getMPdfClassName();

        return new $mPdfClass();
    }

    /**
     * Save PhpWord to file.
     *
     * @param string $filename Name of the file to save as
     */
    public function save($filename = null)
    {
        $fileHandle = parent::prepareForSave($filename);

        //  PDF settings
        $paperSize = strtoupper('A4');
        $orientation = strtoupper('portrait');

        //  Create PDF
        $pdf = $this->createExternalWriterInstance();
        $pdf->_setPageSize($paperSize, $orientation);
        $pdf->addPage($orientation);

        // Write document properties
        $phpWord = $this->getPhpWord();
        $docProps = $phpWord->getDocInfo();
        $pdf->setTitle($docProps->getTitle());
        $pdf->setAuthor($docProps->getCreator());
        $pdf->setSubject($docProps->getSubject());
        $pdf->setKeywords($docProps->getKeywords());
        $pdf->setCreator($docProps->getCreator());

        $pdf->writeHTML($this->getContent());

        //  Write to file
        fwrite($fileHandle, $pdf->output($filename, 'S'));

        parent::restoreStateAfterSave($fileHandle);
    }

    /**
     * Return classname of MPDF to instantiate
     *
     * @codeCoverageIgnore
     * @return string
     */
    private function getMPdfClassName()
    {
        if ($this->includeFile != null) {
            // MPDF version 5.*
            return '\mpdf';
        }

        // MPDF version > 6.*
        return '\Mpdf\Mpdf';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/PDF/TCPDF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PhpWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\PDF;

use PhpOffice\PhpWord\Writer\WriterInterface;

/**
 * TCPDF writer
 *
 * @deprecated 0.13.0 Use `DomPDF` or `MPDF` instead.
 *
 * @see  http://www.tcpdf.org/
 * @since 0.11.0
 */
class TCPDF extends AbstractRenderer implements WriterInterface
{
    /**
     * Name of renderer include file
     *
     * @var string
     */
    protected $includeFile = 'tcpdf.php';

    /**
     * Gets the implementation of external PDF library that should be used.
     *
     * @param string $orientation Page orientation
     * @param string $unit Unit measure
     * @param string $paperSize Paper size
     *
     * @return \TCPDF implementation
     */
    protected function createExternalWriterInstance($orientation, $unit, $paperSize)
    {
        return new \TCPDF($orientation, $unit, $paperSize);
    }

    /**
     * Save PhpWord to file.
     *
     * @param string $filename Name of the file to save as
     */
    public function save($filename = null)
    {
        $fileHandle = parent::prepareForSave($filename);

        //  PDF settings
        $paperSize = 'A4';
        $orientation = 'P';

        // Create PDF
        $pdf = $this->createExternalWriterInstance($orientation, 'pt', $paperSize);
        $pdf->setFontSubsetting(false);
        $pdf->setPrintHeader(false);
        $pdf->setPrintFooter(false);
        $pdf->AddPage();
        $pdf->SetFont($this->getFont());
        $pdf->writeHTML($this->getContent());

        // Write document properties
        $phpWord = $this->getPhpWord();
        $docProps = $phpWord->getDocInfo();
        $pdf->SetTitle($docProps->getTitle());
        $pdf->SetAuthor($docProps->getCreator());
        $pdf->SetSubject($docProps->getSubject());
        $pdf->SetKeywords($docProps->getKeywords());
        $pdf->SetCreator($docProps->getCreator());

        //  Write to file
        fwrite($fileHandle, $pdf->Output($filename, 'S'));

        parent::restoreStateAfterSave($fileHandle);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/PDF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PhpWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Settings;

/**
 * PDF Writer
 *
 * @since 0.10.0
 */
class PDF
{
    /**
     * The wrapper for the requested PDF rendering engine
     *
     * @var \PhpOffice\PhpWord\Writer\PDF\AbstractRenderer
     */
    private $renderer = null;

    /**
     * Instantiate a new renderer of the configured type within this container class
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function __construct(PhpWord $phpWord)
    {
        $pdfLibraryName = Settings::getPdfRendererName();
        $pdfLibraryPath = Settings::getPdfRendererPath();
        if (is_null($pdfLibraryName) || is_null($pdfLibraryPath)) {
            throw new Exception('PDF rendering library or library path has not been defined.');
        }

        $includePath = str_replace('\\', '/', get_include_path());
        $rendererPath = str_replace('\\', '/', $pdfLibraryPath);
        if (strpos($rendererPath, $includePath) === false) {
            set_include_path(get_include_path() . PATH_SEPARATOR . $pdfLibraryPath);
        }

        $rendererName = get_class($this) . '\\' . $pdfLibraryName;
        $this->renderer = new $rendererName($phpWord);
    }

    /**
     * Magic method to handle direct calls to the configured PDF renderer wrapper class.
     *
     * @param string $name Renderer library method name
     * @param mixed[] $arguments Array of arguments to pass to the renderer method
     * @return mixed Returned data from the PDF renderer wrapper method
     */
    public function __call($name, $arguments)
    {
        // Note: Commented because all exceptions should already be catched by `__construct`
        // if ($this->renderer === null) {
        //     throw new Exception("PDF Rendering library has not been defined.");
        // }

        return call_user_func_array(array($this->renderer, $name), $arguments);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/AbstractElement.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

use PhpOffice\PhpWord\Element\AbstractElement as Element;
use PhpOffice\PhpWord\Escaper\Rtf;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font as FontStyle;
use PhpOffice\PhpWord\Style\Paragraph as ParagraphStyle;
use PhpOffice\PhpWord\Writer\AbstractWriter;
use PhpOffice\PhpWord\Writer\HTML\Element\AbstractElement as HTMLAbstractElement;
use PhpOffice\PhpWord\Writer\RTF\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\RTF\Style\Paragraph as ParagraphStyleWriter;

/**
 * Abstract RTF element writer
 *
 * @since 0.11.0
 */
abstract class AbstractElement extends HTMLAbstractElement
{
    /**
     * Font style
     *
     * @var \PhpOffice\PhpWord\Style\Font
     */
    protected $fontStyle;

    /**
     * Paragraph style
     *
     * @var \PhpOffice\PhpWord\Style\Paragraph
     */
    protected $paragraphStyle;

    public function __construct(AbstractWriter $parentWriter, Element $element, $withoutP = false)
    {
        parent::__construct($parentWriter, $element, $withoutP);

        $this->escaper = new Rtf();
    }

    /**
     * Get font and paragraph styles.
     */
    protected function getStyles()
    {
        /** @var \PhpOffice\PhpWord\Writer\RTF $parentWriter Type hint */
        $parentWriter = $this->parentWriter;

        /** @var \PhpOffice\PhpWord\Element\Text $element Type hint */
        $element = $this->element;

        // Font style
        if (method_exists($element, 'getFontStyle')) {
            $this->fontStyle = $element->getFontStyle();
            if (is_string($this->fontStyle)) {
                $this->fontStyle = Style::getStyle($this->fontStyle);
            }
        }

        // Paragraph style
        if (method_exists($element, 'getParagraphStyle')) {
            $this->paragraphStyle = $element->getParagraphStyle();
            if (is_string($this->paragraphStyle)) {
                $this->paragraphStyle = Style::getStyle($this->paragraphStyle);
            }

            if ($this->paragraphStyle !== null && !$this->withoutP) {
                if ($parentWriter->getLastParagraphStyle() != $element->getParagraphStyle()) {
                    $parentWriter->setLastParagraphStyle($element->getParagraphStyle());
                } else {
                    $parentWriter->setLastParagraphStyle();
                    $this->paragraphStyle = null;
                }
            } else {
                $parentWriter->setLastParagraphStyle();
                $this->paragraphStyle = null;
            }
        }
    }

    /**
     * Write opening
     *
     * @return string
     */
    protected function writeOpening()
    {
        if ($this->withoutP || !$this->paragraphStyle instanceof ParagraphStyle) {
            return '';
        }

        $styleWriter = new ParagraphStyleWriter($this->paragraphStyle);
        $styleWriter->setNestedLevel($this->element->getNestedLevel());

        return $styleWriter->write();
    }

    /**
     * Write text
     *
     * @param string $text
     * @return string
     */
    protected function writeText($text)
    {
        if (Settings::isOutputEscapingEnabled()) {
            return $this->escaper->escape($text);
        }

        return SharedText::toUnicode($text); // todo: replace with `return $text;` later.
    }

    /**
     * Write closing
     *
     * @return string
     */
    protected function writeClosing()
    {
        if ($this->withoutP) {
            return '';
        }

        return '\par' . PHP_EOL;
    }

    /**
     * Write font style
     *
     * @return string
     */
    protected function writeFontStyle()
    {
        if (!$this->fontStyle instanceof FontStyle) {
            return '';
        }

        /** @var \PhpOffice\PhpWord\Writer\RTF $parentWriter Type hint */
        $parentWriter = $this->parentWriter;

        // Create style writer and set color/name index
        $styleWriter = new FontStyleWriter($this->fontStyle);
        if ($this->fontStyle->getColor() != null) {
            $colorIndex = array_search($this->fontStyle->getColor(), $parentWriter->getColorTable());
            if ($colorIndex !== false) {
                $styleWriter->setColorIndex($colorIndex + 1);
            }
        }
        if ($this->fontStyle->getName() != null) {
            $fontIndex = array_search($this->fontStyle->getName(), $parentWriter->getFontTable());
            if ($fontIndex !== false) {
                $styleWriter->setNameIndex($fontIndex);
            }
        }

        // Write style
        $content = $styleWriter->write();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Container.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

use PhpOffice\PhpWord\Writer\HTML\Element\Container as HTMLContainer;

/**
 * Container element RTF writer
 *
 * @since 0.11.0
 */
class Container extends HTMLContainer
{
    /**
     * Namespace; Can't use __NAMESPACE__ in inherited class (RTF)
     *
     * @var string
     */
    protected $namespace = 'PhpOffice\\PhpWord\\Writer\\RTF\\Element';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Field.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2019 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * Field element writer
 *
 * Note: for now, only date, page and numpages fields are implemented for RTF.
 */
class Field extends Text
{
    /**
     * Write field element.
     */
    public function write()
    {
        $element = $this->element;
        if (!$element instanceof \PhpOffice\PhpWord\Element\Field) {
            return;
        }

        $this->getStyles();

        $content = '';
        $content .= $this->writeOpening();
        $content .= '{';
        $content .= $this->writeFontStyle();

        $methodName = 'write' . ucfirst(strtolower($element->getType()));
        if (!method_exists($this, $methodName)) {
            // Unsupported field
            $content .= '';
        } else {
            $content .= '\\field{\\*\\fldinst ';
            $content .= $this->$methodName($element);
            $content .= '}{\\fldrslt}';
        }
        $content .= '}';
        $content .= $this->writeClosing();

        return $content;
    }

    protected function writePage()
    {
        return 'PAGE';
    }

    protected function writeNumpages()
    {
        return 'NUMPAGES';
    }

    protected function writeDate(\PhpOffice\PhpWord\Element\Field $element)
    {
        $content = '';
        $content .= 'DATE';
        $properties = $element->getProperties();
        if (isset($properties['dateformat'])) {
            $content .= ' \\\\@ "' . $properties['dateformat'] . '"';
        }

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

use PhpOffice\PhpWord\Element\Image as ImageElement;
use PhpOffice\PhpWord\Shared\Converter;

/**
 * Image element RTF writer
 *
 * @since 0.11.0
 */
class Image extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof ImageElement) {
            return '';
        }

        $this->getStyles();
        $style = $this->element->getStyle();

        $content = '';
        $content .= $this->writeOpening();
        $content .= '{\*\shppict {\pict';
        $content .= '\pngblip\picscalex100\picscaley100';
        $content .= '\picwgoal' . round(Converter::pixelToTwip($style->getWidth()));
        $content .= '\pichgoal' . round(Converter::pixelToTwip($style->getHeight()));
        $content .= PHP_EOL;
        $content .= $this->element->getImageStringData();
        $content .= '}}';
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Link.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * Link element RTF writer
 *
 * @since 0.11.0
 */
class Link extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof \PhpOffice\PhpWord\Element\Link) {
            return '';
        }

        $this->getStyles();

        $content = '';
        $content .= $this->writeOpening();
        $content .= '{\field {\*\fldinst {HYPERLINK "' . $this->element->getSource() . '"}}{\\fldrslt {';
        $content .= $this->writeFontStyle();
        $content .= $this->writeText($this->element->getText());
        $content .= '}}}';
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/ListItem.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * ListItem element RTF writer; extends from text
 *
 * @since 0.11.0
 */
class ListItem extends Text
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/PageBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * PageBreak element RTF writer
 *
 * @since 0.11.0
 */
class PageBreak extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        return '\page' . PHP_EOL;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

use PhpOffice\PhpWord\Element\Cell as CellElement;
use PhpOffice\PhpWord\Element\Row as RowElement;
use PhpOffice\PhpWord\Element\Table as TableElement;

/**
 * Table element RTF writer
 *
 * @since 0.11.0
 */
class Table extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        if (!$this->element instanceof TableElement) {
            return '';
        }
        $element = $this->element;
        // No nesting table for now
        if ($element->getNestedLevel() >= 1) {
            return '';
        }

        $content = '';
        $rows = $element->getRows();
        $rowCount = count($rows);

        if ($rowCount > 0) {
            $content .= '\pard' . PHP_EOL;

            for ($i = 0; $i < $rowCount; $i++) {
                $content .= '\trowd ';
                $content .= $this->writeRowDef($rows[$i]);
                $content .= PHP_EOL;
                $content .= $this->writeRow($rows[$i]);
                $content .= '\row' . PHP_EOL;
            }
            $content .= '\pard' . PHP_EOL;
        }

        return $content;
    }

    /**
     * Write column
     *
     * @param \PhpOffice\PhpWord\Element\Row $row
     * @return string
     */
    private function writeRowDef(RowElement $row)
    {
        $content = '';

        $rightMargin = 0;
        foreach ($row->getCells() as $cell) {
            $width = $cell->getWidth();
            $vMerge = $this->getVMerge($cell->getStyle()->getVMerge());
            if ($width === null) {
                $width = 720; // Arbitrary default width
            }
            $rightMargin += $width;
            $content .= "{$vMerge}\cellx{$rightMargin} ";
        }

        return $content;
    }

    /**
     * Write row
     *
     * @param \PhpOffice\PhpWord\Element\Row $row
     * @return string
     */
    private function writeRow(RowElement $row)
    {
        $content = '';

        // Write cells
        foreach ($row->getCells() as $cell) {
            $content .= $this->writeCell($cell);
        }

        return $content;
    }

    /**
     * Write cell
     *
     * @param \PhpOffice\PhpWord\Element\Cell $cell
     * @return string
     */
    private function writeCell(CellElement $cell)
    {
        $content = '\intbl' . PHP_EOL;

        // Write content
        $writer = new Container($this->parentWriter, $cell);
        $content .= $writer->write();

        $content .= '\cell' . PHP_EOL;

        return $content;
    }

    /**
     * Get vertical merge style
     *
     * @param string $value
     * @return string
     * @todo Move to style
     */
    private function getVMerge($value)
    {
        $style = '';
        if ($value == 'restart') {
            $style = '\clvmgf';
        } elseif ($value == 'continue') {
            $style = '\clvmrg';
        }

        return $style;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * Text element RTF writer
 *
 * @since 0.10.0
 */
class Text extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Element\Text $element Type hint */
        $element = $this->element;
        $elementClass = str_replace('\\Writer\\RTF', '', get_class($this));
        if (!$element instanceof $elementClass || !is_string($element->getText())) {
            return '';
        }

        $this->getStyles();

        $content = '';
        $content .= $this->writeOpening();
        $content .= '{';
        $content .= $this->writeFontStyle();
        $content .= $this->writeText($element->getText());
        $content .= '}';
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/TextBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * TextBreak element RTF writer
 *
 * @since 0.10.0
 */
class TextBreak extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Writer\RTF $parentWriter Type hint */
        $parentWriter = $this->parentWriter;
        $parentWriter->setLastParagraphStyle();

        return '\pard\par' . PHP_EOL;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/TextRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * TextRun element RTF writer
 *
 * @since 0.10.0
 */
class TextRun extends AbstractElement
{
    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        $writer = new Container($this->parentWriter, $this->element);
        $this->getStyles();

        $content = '';
        $content .= $this->writeOpening();
        $content .= '{';
        $content .= $writer->write();
        $content .= '}';
        $content .= $this->writeClosing();

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Element/Title.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Element;

/**
 * Title element RTF writer; extends from text
 *
 * @since 0.11.0
 */
class Title extends Text
{
    protected function getStyles()
    {
        /** @var \PhpOffice\PhpWord\Element\Title $element Type hint */
        $element = $this->element;
        $style = $element->getStyle();
        $style = str_replace('Heading', 'Heading_', $style);
        $style = \PhpOffice\PhpWord\Style::getStyle($style);
        if ($style instanceof \PhpOffice\PhpWord\Style\Font) {
            $this->fontStyle = $style;
            $pstyle = $style->getParagraph();
            if ($pstyle instanceof \PhpOffice\PhpWord\Style\Paragraph && $pstyle->hasPageBreakBefore()) {
                $sect = $element->getParent();
                if ($sect instanceof \PhpOffice\PhpWord\Element\Section) {
                    $elems = $sect->getElements();
                    if ($elems[0] === $element) {
                        $pstyle = clone $pstyle;
                        $pstyle->setPageBreakBefore(false);
                    }
                }
            }
            $this->paragraphStyle = $pstyle;
        }
    }

    /**
     * Write element
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Element\Title $element Type hint */
        $element = $this->element;
        $elementClass = str_replace('\\Writer\\RTF', '', get_class($this));
        if (!$element instanceof $elementClass || !is_string($element->getText())) {
            return '';
        }

        $this->getStyles();

        $content = '';

        $content .= $this->writeOpening();
        $endout = '';
        $style = $element->getStyle();
        if (is_string($style)) {
            $style = str_replace('Heading', '', $style);
            if (is_numeric($style)) {
                $style = (int) $style - 1;
                if ($style >= 0 && $style <= 8) {
                    $content .= '{\\outlinelevel' . $style;
                    $endout = '}';
                }
            }
        }

        $content .= '{';
        $content .= $this->writeFontStyle();
        $content .= $this->writeText($element->getText());
        $content .= '}';
        $content .= $this->writeClosing();
        $content .= $endout;

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Part/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Part;

use PhpOffice\PhpWord\Escaper\Rtf;
use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Writer\AbstractWriter;

/**
 * @since 0.11.0
 */
abstract class AbstractPart
{
    /**
     * @var \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    private $parentWriter;

    /**
     * @var \PhpOffice\PhpWord\Escaper\EscaperInterface
     */
    protected $escaper;

    public function __construct()
    {
        $this->escaper = new Rtf();
    }

    /**
     * @return string
     */
    abstract public function write();

    /**
     * @param \PhpOffice\PhpWord\Writer\AbstractWriter $writer
     */
    public function setParentWriter(AbstractWriter $writer = null)
    {
        $this->parentWriter = $writer;
    }

    /**
     * @throws \PhpOffice\PhpWord\Exception\Exception
     * @return \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    public function getParentWriter()
    {
        if ($this->parentWriter !== null) {
            return $this->parentWriter;
        }
        throw new Exception('No parent WriterInterface assigned.');
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Part/Document.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Part;

use PhpOffice\PhpWord\Element\Footer;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Writer\RTF\Element\Container;
use PhpOffice\PhpWord\Writer\RTF\Style\Section as SectionStyleWriter;

/**
 * RTF document part writer
 *
 * @since 0.11.0
 * @see  http://www.biblioscape.com/rtf15_spec.htm#Heading24
 */
class Document extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $content = '';

        $content .= $this->writeInfo();
        $content .= $this->writeFormatting();
        $content .= $this->writeSections();

        return $content;
    }

    /**
     * Write document information
     *
     * @return string
     */
    private function writeInfo()
    {
        $docProps = $this->getParentWriter()->getPhpWord()->getDocInfo();
        $properties = array('title', 'subject', 'category', 'keywords', 'comment',
            'author', 'operator', 'creatim', 'revtim', 'company', 'manager', );
        $mapping = array(
            'comment'  => 'description',
            'author'   => 'creator',
            'operator' => 'lastModifiedBy',
            'creatim'  => 'created',
            'revtim'   => 'modified', );
        $dateFields = array('creatim', 'revtim');

        $content = '';

        $content .= '{';
        $content .= '\info';
        foreach ($properties as $property) {
            $method = 'get' . (isset($mapping[$property]) ? $mapping[$property] : $property);
            if (!in_array($property, $dateFields) && Settings::isOutputEscapingEnabled()) {
                $value = $this->escaper->escape($docProps->$method());
            } else {
                $value = $docProps->$method();
            }
            $value = in_array($property, $dateFields) ? $this->getDateValue($value) : $value;
            $content .= "{\\{$property} {$value}}";
        }
        $content .= '}';
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write document formatting properties
     *
     * @return string
     */
    private function writeFormatting()
    {
        $docSettings = $this->getParentWriter()->getPhpWord()->getSettings();
        // Applies a language to a text run (defaults to 1036 : French (France))
        $langId = $docSettings->getThemeFontLang() != null && $docSettings->getThemeFontLang()->getLangId() != null ? $docSettings->getThemeFontLang()->getLangId() : 1036;

        $content = '';

        $content .= '\deftab720'; // Set the default tab size (720 twips)
        $content .= '\viewkind1'; // Set the view mode of the document

        $content .= '\uc1'; // Set the numberof bytes that follows a unicode character
        $content .= '\pard'; // Resets to default paragraph properties.
        $content .= '\nowidctlpar'; // No widow/orphan control
        $content .= '\lang' . $langId;
        $content .= '\kerning1'; // Point size (in half-points) above which to kern character pairs
        $content .= '\fs' . (Settings::getDefaultFontSize() * 2); // Set the font size in half-points
        if ($docSettings->hasEvenAndOddHeaders()) {
            $content .= '\\facingp';
        }
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write titlepg directive if any "f" headers or footers
     *
     * @param \PhpOffice\PhpWord\Element\Section $section
     * @return string
     */
    private static function writeTitlepg($section)
    {
        foreach ($section->getHeaders() as $header) {
            if ($header->getType() === Footer::FIRST) {
                return '\\titlepg' . PHP_EOL;
            }
        }
        foreach ($section->getFooters() as $header) {
            if ($header->getType() === Footer::FIRST) {
                return '\\titlepg' . PHP_EOL;
            }
        }

        return '';
    }

    /**
     * Write sections
     *
     * @return string
     */
    private function writeSections()
    {
        $content = '';

        $sections = $this->getParentWriter()->getPhpWord()->getSections();
        $evenOdd = $this->getParentWriter()->getPhpWord()->getSettings()->hasEvenAndOddHeaders();
        foreach ($sections as $section) {
            $styleWriter = new SectionStyleWriter($section->getStyle());
            $styleWriter->setParentWriter($this->getParentWriter());
            $content .= $styleWriter->write();
            $content .= self::writeTitlepg($section);

            foreach ($section->getHeaders() as $header) {
                $type = $header->getType();
                if ($evenOdd || $type !== FOOTER::EVEN) {
                    $content .= '{\\header';
                    if ($type === Footer::FIRST) {
                        $content .= 'f';
                    } elseif ($evenOdd) {
                        $content .= ($type === FOOTER::EVEN) ? 'l' : 'r';
                    }
                    foreach ($header->getElements() as $element) {
                        $cl = get_class($element);
                        $cl2 = str_replace('Element', 'Writer\\RTF\\Element', $cl);
                        if (class_exists($cl2)) {
                            $elementWriter = new $cl2($this->getParentWriter(), $element);
                            $content .= $elementWriter->write();
                        }
                    }
                    $content .= '}' . PHP_EOL;
                }
            }
            foreach ($section->getFooters() as $footer) {
                $type = $footer->getType();
                if ($evenOdd || $type !== FOOTER::EVEN) {
                    $content .= '{\\footer';
                    if ($type === Footer::FIRST) {
                        $content .= 'f';
                    } elseif ($evenOdd) {
                        $content .= ($type === FOOTER::EVEN) ? 'l' : 'r';
                    }
                    foreach ($footer->getElements() as $element) {
                        $cl = get_class($element);
                        $cl2 = str_replace('Element', 'Writer\\RTF\\Element', $cl);
                        if (class_exists($cl2)) {
                            $elementWriter = new $cl2($this->getParentWriter(), $element);
                            $content .= $elementWriter->write();
                        }
                    }
                    $content .= '}' . PHP_EOL;
                }
            }

            $elementWriter = new Container($this->getParentWriter(), $section);
            $content .= $elementWriter->write();

            $content .= '\sect' . PHP_EOL;
        }

        return $content;
    }

    /**
     * Get date value
     *
     * The format of date value is `\yr?\mo?\dy?\hr?\min?\sec?`
     *
     * @param int $value
     * @return string
     */
    private function getDateValue($value)
    {
        $dateParts = array(
            'Y' => 'yr',
            'm' => 'mo',
            'd' => 'dy',
            'H' => 'hr',
            'i' => 'min',
            's' => 'sec',
        );
        $result = '';
        foreach ($dateParts as $dateFormat => $controlWord) {
            $result .= '\\' . $controlWord . date($dateFormat, $value);
        }

        return $result;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Part/Header.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Part;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\Converter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font;

/**
 * RTF header part writer
 *
 * - Character set
 * - Font table
 * - File table (not supported yet)
 * - Color table
 * - Style sheet (not supported yet)
 * - List table (not supported yet)
 *
 * @since 0.11.0
 * @see  http://www.biblioscape.com/rtf15_spec.htm#Heading6
 */
class Header extends AbstractPart
{
    /**
     * Font table
     *
     * @var array
     */
    private $fontTable = array();

    /**
     * Color table
     *
     * @var array
     */
    private $colorTable = array();

    /**
     * Get font table.
     *
     * @return array
     */
    public function getFontTable()
    {
        return $this->fontTable;
    }

    /**
     * Get color table.
     *
     * @return array
     */
    public function getColorTable()
    {
        return $this->colorTable;
    }

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $this->registerFont();

        $content = '';

        $content .= $this->writeCharset();
        $content .= $this->writeDefaults();
        $content .= $this->writeFontTable();
        $content .= $this->writeColorTable();
        $content .= $this->writeGenerator();
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write character set
     *
     * @return string
     */
    private function writeCharset()
    {
        $content = '';

        $content .= '\ansi';
        $content .= '\ansicpg1252';
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write header defaults
     *
     * @return string
     */
    private function writeDefaults()
    {
        $content = '';

        $content .= '\deff0';
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write font table
     *
     * @return string
     */
    private function writeFontTable()
    {
        $content = '';

        $content .= '{';
        $content .= '\fonttbl';
        foreach ($this->fontTable as $index => $font) {
            $content .= "{\\f{$index}\\fnil\\fcharset0 {$font};}";
        }
        $content .= '}';
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write color table
     *
     * @return string
     */
    private function writeColorTable()
    {
        $content = '';

        $content .= '{';
        $content .= '\colortbl;';
        foreach ($this->colorTable as $color) {
            list($red, $green, $blue) = Converter::htmlToRgb($color);
            $content .= "\\red{$red}\\green{$green}\\blue{$blue};";
        }
        $content .= '}';
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Write
     *
     * @return string
     */
    private function writeGenerator()
    {
        $content = '';

        $content .= '{\*\generator PHPWord;}'; // Set the generator
        $content .= PHP_EOL;

        return $content;
    }

    /**
     * Register all fonts and colors in both named and inline styles to appropriate header table.
     */
    private function registerFont()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $this->fontTable[] = Settings::getDefaultFontName();

        // Search named styles
        $styles = Style::getStyles();
        foreach ($styles as $style) {
            $this->registerFontItems($style);
        }

        // Search inline styles
        $sections = $phpWord->getSections();
        foreach ($sections as $section) {
            $elements = $section->getElements();
            $this->registerBorderColor($section->getStyle());
            foreach ($elements as $element) {
                if (method_exists($element, 'getFontStyle')) {
                    $style = $element->getFontStyle();
                    $this->registerFontItems($style);
                }
            }
        }
    }

    /**
     * Register border colors.
     *
     * @param \PhpOffice\PhpWord\Style\Border $style
     */
    private function registerBorderColor($style)
    {
        $colors = $style->getBorderColor();
        foreach ($colors as $color) {
            if ($color !== null) {
                $this->registerTableItem($this->colorTable, $color);
            }
        }
    }

    /**
     * Register fonts and colors.
     *
     * @param \PhpOffice\PhpWord\Style\AbstractStyle $style
     */
    private function registerFontItems($style)
    {
        $defaultFont = Settings::getDefaultFontName();
        $defaultColor = Settings::DEFAULT_FONT_COLOR;

        if ($style instanceof Font) {
            $this->registerTableItem($this->fontTable, $style->getName(), $defaultFont);
            $this->registerTableItem($this->colorTable, $style->getColor(), $defaultColor);
            $this->registerTableItem($this->colorTable, $style->getFgColor(), $defaultColor);
        }
    }

    /**
     * Register individual font and color.
     *
     * @param array &$table
     * @param string $value
     * @param string $default
     */
    private function registerTableItem(&$table, $value, $default = null)
    {
        if (in_array($value, $table) === false && $value !== null && $value != $default) {
            $table[] = $value;
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/AbstractStyle.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

use PhpOffice\PhpWord\Writer\HTML\Style\AbstractStyle as HTMLAbstractStyle;

/**
 * Abstract RTF style writer
 *
 * @since 0.11.0
 */
abstract class AbstractStyle extends HTMLAbstractStyle
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Border.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

/**
 * Border style writer
 *
 * @since 0.12.0
 */
class Border extends AbstractStyle
{
    /**
     * Sizes
     *
     * @var array
     */
    private $sizes = array();

    /**
     * Colors
     *
     * @var array
     */
    private $colors = array();

    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $content = '';

        $sides = array('top', 'left', 'right', 'bottom');
        $sizeCount = count($this->sizes);

        // Page border measure
        // 8 = from text, infront off; 32 = from edge, infront on; 40 = from edge, infront off
        $content .= '\pgbrdropt32';

        for ($i = 0; $i < $sizeCount; $i++) {
            if ($this->sizes[$i] !== null) {
                $color = null;
                if (isset($this->colors[$i])) {
                    $color = $this->colors[$i];
                }
                $content .= $this->writeSide($sides[$i], $this->sizes[$i], $color);
            }
        }

        return $content;
    }

    /**
     * Write side
     *
     * @param string $side
     * @param int $width
     * @param string $color
     * @return string
     */
    private function writeSide($side, $width, $color = '')
    {
        /** @var \PhpOffice\PhpWord\Writer\RTF $rtfWriter */
        $rtfWriter = $this->getParentWriter();
        $colorIndex = 0;
        if ($rtfWriter !== null) {
            $colorTable = $rtfWriter->getColorTable();
            $index = array_search($color, $colorTable);
            if ($index !== false && $colorIndex !== null) {
                $colorIndex = $index + 1;
            }
        }

        $content = '';

        $content .= '\pgbrdr' . substr($side, 0, 1);
        $content .= '\brdrs'; // Single-thickness border; @todo Get other type of border
        $content .= '\brdrw' . round($width); // Width
        $content .= '\brdrcf' . $colorIndex; // Color
        $content .= '\brsp480'; // Space in twips between borders and the paragraph (24pt, following OOXML)
        $content .= ' ';

        return $content;
    }

    /**
     * Set sizes.
     *
     * @param int[] $value
     */
    public function setSizes($value)
    {
        $this->sizes = $value;
    }

    /**
     * Set colors.
     *
     * @param string[] $value
     */
    public function setColors($value)
    {
        $this->colors = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Font.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

use PhpOffice\PhpWord\Style\Font as FontStyle;

/**
 * RTF font style writer
 *
 * @since 0.11.0
 */
class Font extends AbstractStyle
{
    /**
     * @var int Font name index
     */
    private $nameIndex = 0;

    /**
     * @var int Font color index
     */
    private $colorIndex = 0;

    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof FontStyle) {
            return '';
        }

        $content = '';
        $content .= $this->getValueIf($style->isRTL(), '\rtlch');
        $content .= '\cf' . $this->colorIndex;
        $content .= '\f' . $this->nameIndex;

        $size = $style->getSize();
        $content .= $this->getValueIf(is_numeric($size), '\fs' . round($size * 2));

        $content .= $this->getValueIf($style->isBold(), '\b');
        $content .= $this->getValueIf($style->isItalic(), '\i');
        $content .= $this->getValueIf($style->getUnderline() != FontStyle::UNDERLINE_NONE, '\ul');
        $content .= $this->getValueIf($style->isStrikethrough(), '\strike');
        $content .= $this->getValueIf($style->isSuperScript(), '\super');
        $content .= $this->getValueIf($style->isSubScript(), '\sub');

        return $content . ' ';
    }

    /**
     * Set font name index.
     *
     *
     * @param int $value
     */
    public function setNameIndex($value = 0)
    {
        $this->nameIndex = $value;
    }

    /**
     * Set font color index.
     *
     * @param int $value
     */
    public function setColorIndex($value = 0)
    {
        $this->colorIndex = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Indentation.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

/**
 * RTF indentation style writer
 *
 * @since 0.11.0
 */
class Indentation extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Indentation) {
            return '';
        }

        $content = '\fi' . round($style->getFirstLine());
        $content .= '\li' . round($style->getLeft());
        $content .= '\ri' . round($style->getRight());

        return $content . ' ';
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Paragraph.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

use PhpOffice\PhpWord\SimpleType\Jc;

/**
 * RTF paragraph style writer
 *
 * @since 0.11.0
 */
class Paragraph extends AbstractStyle
{
    /**
     * Depth of table container nested level; Primarily used for RTF writer/reader
     *
     * 0 = Not in a table; 1 = in a table; 2 = in a table inside another table, etc.
     *
     * @var int
     */
    private $nestedLevel = 0;

    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Paragraph) {
            return '';
        }

        $alignments = array(
            Jc::START  => '\ql',
            Jc::END    => '\qr',
            Jc::CENTER => '\qc',
            Jc::BOTH   => '\qj',
        );

        $spaceAfter = $style->getSpaceAfter();
        $spaceBefore = $style->getSpaceBefore();

        $content = '';
        if ($this->nestedLevel == 0) {
            $content .= '\pard\nowidctlpar ';
        }
        if (isset($alignments[$style->getAlignment()])) {
            $content .= $alignments[$style->getAlignment()];
        }
        $content .= $this->writeIndentation($style->getIndentation());
        $content .= $this->getValueIf($spaceBefore !== null, '\sb' . round($spaceBefore));
        $content .= $this->getValueIf($spaceAfter !== null, '\sa' . round($spaceAfter));
        $lineHeight = $style->getLineHeight();
        if ($lineHeight) {
            $lineHeightAdjusted = (int) ($lineHeight * 240);
            $content .= "\\sl$lineHeightAdjusted\\slmult1";
        }
        if ($style->hasPageBreakBefore()) {
            $content .= '\\page';
        }

        $styles = $style->getStyleValues();
        $content .= $this->writeTabs($styles['tabs']);

        return $content;
    }

    /**
     * Writes an \PhpOffice\PhpWord\Style\Indentation
     *
     * @param null|\PhpOffice\PhpWord\Style\Indentation $indent
     * @return string
     */
    private function writeIndentation($indent = null)
    {
        if (isset($indent) && $indent instanceof \PhpOffice\PhpWord\Style\Indentation) {
            $writer = new Indentation($indent);

            return $writer->write();
        }

        return '';
    }

    /**
     * Writes tabs
     *
     * @param \PhpOffice\PhpWord\Style\Tab[] $tabs
     * @return string
     */
    private function writeTabs($tabs = null)
    {
        $content = '';
        if (!empty($tabs)) {
            foreach ($tabs as $tab) {
                $styleWriter = new Tab($tab);
                $content .= $styleWriter->write();
            }
        }

        return $content;
    }

    /**
     * Set nested level.
     *
     * @param int $value
     */
    public function setNestedLevel($value)
    {
        $this->nestedLevel = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Section.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

use PhpOffice\PhpWord\Style\Section as SectionStyle;

/**
 * RTF section style writer
 *
 * @since 0.12.0
 */
class Section extends AbstractStyle
{
    /**
     * Write style
     *
     * @return string
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof SectionStyle) {
            return '';
        }

        $content = '';

        $content .= '\sectd ';

        // Size & margin
        $content .= $this->getValueIf($style->getPageSizeW() !== null, '\pgwsxn' . round($style->getPageSizeW()));
        $content .= $this->getValueIf($style->getPageSizeH() !== null, '\pghsxn' . round($style->getPageSizeH()));
        $content .= ' ';
        $content .= $this->getValueIf($style->getMarginTop() !== null, '\margtsxn' . round($style->getMarginTop()));
        $content .= $this->getValueIf($style->getMarginRight() !== null, '\margrsxn' . round($style->getMarginRight()));
        $content .= $this->getValueIf($style->getMarginBottom() !== null, '\margbsxn' . round($style->getMarginBottom()));
        $content .= $this->getValueIf($style->getMarginLeft() !== null, '\marglsxn' . round($style->getMarginLeft()));
        $content .= $this->getValueIf($style->getHeaderHeight() !== null, '\headery' . round($style->getHeaderHeight()));
        $content .= $this->getValueIf($style->getFooterHeight() !== null, '\footery' . round($style->getFooterHeight()));
        $content .= $this->getValueIf($style->getGutter() !== null, '\guttersxn' . round($style->getGutter()));
        $content .= $this->getValueIf($style->getPageNumberingStart() !== null, '\pgnstarts' . $style->getPageNumberingStart() . '\pgnrestart');
        $content .= ' ';

        // Borders
        if ($style->hasBorder()) {
            $styleWriter = new Border($style);
            $styleWriter->setParentWriter($this->getParentWriter());
            $styleWriter->setSizes($style->getBorderSize());
            $styleWriter->setColors($style->getBorderColor());
            $content .= $styleWriter->write();
        }

        return $content . PHP_EOL;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF/Style/Tab.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\RTF\Style;

/**
 * Line numbering style writer
 *
 * @since 0.10.0
 */
class Tab extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Tab) {
            return;
        }
        $tabs = array(
            \PhpOffice\PhpWord\Style\Tab::TAB_STOP_RIGHT   => '\tqr',
            \PhpOffice\PhpWord\Style\Tab::TAB_STOP_CENTER  => '\tqc',
            \PhpOffice\PhpWord\Style\Tab::TAB_STOP_DECIMAL => '\tqdec',
        );
        $content = '';
        if (isset($tabs[$style->getType()])) {
            $content .= $tabs[$style->getType()];
        }
        $content .= '\tx' . round($style->getPosition());

        return $content;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/RTF.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\PhpWord;

/**
 * RTF writer
 *
 * @since 0.7.0
 */
class RTF extends AbstractWriter implements WriterInterface
{
    /**
     * Last paragraph style
     *
     * @var mixed
     */
    private $lastParagraphStyle;

    /**
     * Create new instance
     *
     * @param \PhpOffice\PhpWord\PhpWord $phpWord
     */
    public function __construct(PhpWord $phpWord = null)
    {
        $this->setPhpWord($phpWord);

        $this->parts = array('Header', 'Document');
        foreach ($this->parts as $partName) {
            $partClass = get_class($this) . '\\Part\\' . $partName;
            if (class_exists($partClass)) {
                /** @var \PhpOffice\PhpWord\Writer\RTF\Part\AbstractPart $part Type hint */
                $part = new $partClass();
                $part->setParentWriter($this);
                $this->writerParts[strtolower($partName)] = $part;
            }
        }
    }

    /**
     * Save content to file.
     *
     * @param string $filename
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    public function save($filename = null)
    {
        $this->writeFile($this->openFile($filename), $this->getContent());
    }

    /**
     * Get content
     *
     * @return string
     * @since 0.11.0
     */
    private function getContent()
    {
        $content = '';

        $content .= '{';
        $content .= '\rtf1' . PHP_EOL;
        $content .= $this->getWriterPart('Header')->write();
        $content .= $this->getWriterPart('Document')->write();
        $content .= '}';

        return $content;
    }

    /**
     * Get font table.
     *
     * @return array
     */
    public function getFontTable()
    {
        return $this->getWriterPart('Header')->getFontTable();
    }

    /**
     * Get color table.
     *
     * @return array
     */
    public function getColorTable()
    {
        return $this->getWriterPart('Header')->getColorTable();
    }

    /**
     * Get last paragraph style.
     *
     * @return mixed
     */
    public function getLastParagraphStyle()
    {
        return $this->lastParagraphStyle;
    }

    /**
     * Set last paragraph style.
     *
     * @param mixed $value
     */
    public function setLastParagraphStyle($value = '')
    {
        $this->lastParagraphStyle = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/AbstractElement.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\AbstractElement as Element;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\Text as SharedText;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Abstract element writer
 *
 * @since 0.11.0
 */
abstract class AbstractElement
{
    /**
     * XML writer
     *
     * @var \PhpOffice\PhpWord\Shared\XMLWriter
     */
    private $xmlWriter;

    /**
     * Element
     *
     * @var \PhpOffice\PhpWord\Element\AbstractElement
     */
    private $element;

    /**
     * Without paragraph
     *
     * @var bool
     */
    protected $withoutP = false;

    /**
     * Write element
     */
    abstract public function write();

    /**
     * Create new instance
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\AbstractElement $element
     * @param bool $withoutP
     */
    public function __construct(XMLWriter $xmlWriter, Element $element, $withoutP = false)
    {
        $this->xmlWriter = $xmlWriter;
        $this->element = $element;
        $this->withoutP = $withoutP;
    }

    /**
     * Get XML Writer
     *
     * @return \PhpOffice\PhpWord\Shared\XMLWriter
     */
    protected function getXmlWriter()
    {
        return $this->xmlWriter;
    }

    /**
     * Get element
     *
     * @return \PhpOffice\PhpWord\Element\AbstractElement
     */
    protected function getElement()
    {
        return $this->element;
    }

    /**
     * Start w:p DOM element.
     *
     * @uses \PhpOffice\PhpWord\Writer\Word2007\Element\PageBreak::write()
     */
    protected function startElementP()
    {
        if (!$this->withoutP) {
            $this->xmlWriter->startElement('w:p');
            // Paragraph style
            if (method_exists($this->element, 'getParagraphStyle')) {
                $this->writeParagraphStyle();
            }
        }
        $this->writeCommentRangeStart();
    }

    /**
     * End w:p DOM element.
     */
    protected function endElementP()
    {
        $this->writeCommentRangeEnd();
        if (!$this->withoutP) {
            $this->xmlWriter->endElement(); // w:p
        }
    }

    /**
     * Writes the w:commentRangeStart DOM element
     */
    protected function writeCommentRangeStart()
    {
        if ($this->element->getCommentRangeStart() != null) {
            $comment = $this->element->getCommentRangeStart();
            //only set the ID if it is not yet set, otherwise it will overwrite it
            if ($comment->getElementId() == null) {
                $comment->setElementId();
            }

            $this->xmlWriter->writeElementBlock('w:commentRangeStart', array('w:id' => $comment->getElementId()));
        }
    }

    /**
     * Writes the w:commentRangeEnd DOM element
     */
    protected function writeCommentRangeEnd()
    {
        if ($this->element->getCommentRangeEnd() != null) {
            $comment = $this->element->getCommentRangeEnd();
            //only set the ID if it is not yet set, otherwise it will overwrite it, this should normally not happen
            if ($comment->getElementId() == null) {
                $comment->setElementId(); // @codeCoverageIgnore
            } // @codeCoverageIgnore

            $this->xmlWriter->writeElementBlock('w:commentRangeEnd', array('w:id' => $comment->getElementId()));
            $this->xmlWriter->startElement('w:r');
            $this->xmlWriter->writeElementBlock('w:commentReference', array('w:id' => $comment->getElementId()));
            $this->xmlWriter->endElement();
        } elseif ($this->element->getCommentRangeStart() != null && $this->element->getCommentRangeStart()->getEndElement() == null) {
            $comment = $this->element->getCommentRangeStart();
            //only set the ID if it is not yet set, otherwise it will overwrite it, this should normally not happen
            if ($comment->getElementId() == null) {
                $comment->setElementId(); // @codeCoverageIgnore
            } // @codeCoverageIgnore

            $this->xmlWriter->writeElementBlock('w:commentRangeEnd', array('w:id' => $comment->getElementId()));
            $this->xmlWriter->startElement('w:r');
            $this->xmlWriter->writeElementBlock('w:commentReference', array('w:id' => $comment->getElementId()));
            $this->xmlWriter->endElement();
        }
    }

    /**
     * Write ending.
     */
    protected function writeParagraphStyle()
    {
        $this->writeTextStyle('Paragraph');
    }

    /**
     * Write ending.
     */
    protected function writeFontStyle()
    {
        $this->writeTextStyle('Font');
    }

    /**
     * Write text style.
     *
     * @param string $styleType Font|Paragraph
     */
    private function writeTextStyle($styleType)
    {
        $method = "get{$styleType}Style";
        $class = "PhpOffice\\PhpWord\\Writer\\Word2007\\Style\\{$styleType}";
        $styleObject = $this->element->$method();

        /** @var \PhpOffice\PhpWord\Writer\Word2007\Style\AbstractStyle $styleWriter Type Hint */
        $styleWriter = new $class($this->xmlWriter, $styleObject);
        if (method_exists($styleWriter, 'setIsInline')) {
            $styleWriter->setIsInline(true);
        }

        $styleWriter->write();
    }

    /**
     * Convert text to valid format
     *
     * @param string $text
     * @return string
     */
    protected function getText($text)
    {
        return SharedText::controlCharacterPHP2OOXML($text);
    }

    /**
     * Write an XML text, this will call text() or writeRaw() depending on the value of Settings::isOutputEscapingEnabled()
     *
     * @param string $content The text string to write
     * @return bool Returns true on success or false on failure
     */
    protected function writeText($content)
    {
        if (Settings::isOutputEscapingEnabled()) {
            return $this->getXmlWriter()->text($content);
        }

        return $this->getXmlWriter()->writeRaw($content);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Bookmark.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * Bookmark element writer
 *
 * @since 0.12.0
 */
class Bookmark extends AbstractElement
{
    /**
     * Write bookmark element
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Bookmark) {
            return;
        }

        $rId = $element->getRelationId();

        $xmlWriter->startElement('w:bookmarkStart');
        $xmlWriter->writeAttribute('w:id', $rId);
        $xmlWriter->writeAttribute('w:name', $element->getName());
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:bookmarkEnd');
        $xmlWriter->writeAttribute('w:id', $rId);
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Chart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\Chart as ChartElement;

/**
 * Chart element writer
 *
 * @since 0.12.0
 */
class Chart extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof ChartElement) {
            return;
        }

        $rId = $element->getRelationId();
        $style = $element->getStyle();

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:drawing');
        $xmlWriter->startElement('wp:inline');

        // EMU
        $xmlWriter->writeElementBlock('wp:extent', array('cx' => $style->getWidth(), 'cy' => $style->getHeight()));
        $xmlWriter->writeElementBlock('wp:docPr', array('id' => $rId, 'name' => "Chart{$rId}"));

        $xmlWriter->startElement('a:graphic');
        $xmlWriter->writeAttribute('xmlns:a', 'http://schemas.openxmlformats.org/drawingml/2006/main');
        $xmlWriter->startElement('a:graphicData');
        $xmlWriter->writeAttribute('uri', 'http://schemas.openxmlformats.org/drawingml/2006/chart');

        $xmlWriter->startElement('c:chart');
        $xmlWriter->writeAttribute('r:id', "rId{$rId}");
        $xmlWriter->writeAttribute('xmlns:c', 'http://schemas.openxmlformats.org/drawingml/2006/chart');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->endElement(); // c:chart

        $xmlWriter->endElement(); // a:graphicData
        $xmlWriter->endElement(); // a:graphic

        $xmlWriter->endElement(); // wp:inline
        $xmlWriter->endElement(); // w:drawing
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/CheckBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * CheckBox element writer
 *
 * @since 0.10.0
 */
class CheckBox extends Text
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\CheckBox) {
            return;
        }

        $this->startElementP();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->startElement('w:ffData');
        $xmlWriter->startElement('w:name');
        $xmlWriter->writeAttribute('w:val', $this->getText($element->getName()));
        $xmlWriter->endElement(); //w:name
        $xmlWriter->writeAttribute('w:enabled', '');
        $xmlWriter->startElement('w:calcOnExit');
        $xmlWriter->writeAttribute('w:val', '0');
        $xmlWriter->endElement(); //w:calcOnExit
        $xmlWriter->startElement('w:checkBox');
        $xmlWriter->writeAttribute('w:sizeAuto', '');
        $xmlWriter->startElement('w:default');
        $xmlWriter->writeAttribute('w:val', 0);
        $xmlWriter->endElement(); //w:default
        $xmlWriter->endElement(); //w:checkBox
        $xmlWriter->endElement(); // w:ffData
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text(' FORMCHECKBOX ');
        $xmlWriter->endElement(); // w:instrText
        $xmlWriter->endElement(); // w:r
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'separate');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'end');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');

        $this->writeFontStyle();

        $xmlWriter->startElement('w:t');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $this->writeText($this->getText($element->getText()));
        $xmlWriter->endElement(); // w:t
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Container.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\AbstractContainer as ContainerElement;
use PhpOffice\PhpWord\Element\AbstractElement as Element;
use PhpOffice\PhpWord\Element\TextBreak as TextBreakElement;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Container element writer (section, textrun, header, footnote, cell, etc.)
 *
 * @since 0.11.0
 */
class Container extends AbstractElement
{
    /**
     * Namespace; Can't use __NAMESPACE__ in inherited class (ODText)
     *
     * @var string
     */
    protected $namespace = 'PhpOffice\\PhpWord\\Writer\\Word2007\\Element';

    /**
     * Write element.
     */
    public function write()
    {
        $container = $this->getElement();
        if (!$container instanceof ContainerElement) {
            return;
        }
        $containerClass = substr(get_class($container), strrpos(get_class($container), '\\') + 1);
        $withoutP = in_array($containerClass, array('TextRun', 'Footnote', 'Endnote', 'ListItemRun'));
        $xmlWriter = $this->getXmlWriter();

        // Loop through elements
        $elements = $container->getElements();
        $elementClass = '';
        foreach ($elements as $element) {
            $elementClass = $this->writeElement($xmlWriter, $element, $withoutP);
        }

        // Special case for Cell: They have to contain a w:p element at the end.
        // The $elementClass contains the last element name. If it's empty string
        // or Table, the last element is not w:p
        $writeLastTextBreak = ($containerClass == 'Cell') && ($elementClass == '' || $elementClass == 'Table');
        if ($writeLastTextBreak) {
            $writerClass = $this->namespace . '\\TextBreak';
            /** @var \PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement $writer Type hint */
            $writer = new $writerClass($xmlWriter, new TextBreakElement(), $withoutP);
            $writer->write();
        }
    }

    /**
     * Write individual element
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\AbstractElement $element
     * @param bool $withoutP
     * @return string
     */
    private function writeElement(XMLWriter $xmlWriter, Element $element, $withoutP)
    {
        $elementClass = substr(get_class($element), strrpos(get_class($element), '\\') + 1);
        $writerClass = $this->namespace . '\\' . $elementClass;

        if (class_exists($writerClass)) {
            /** @var \PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement $writer Type hint */
            $writer = new $writerClass($xmlWriter, $element, $withoutP);
            $writer->write();
        }

        return $elementClass;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Endnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * Endnote element writer
 *
 * @since 0.10.0
 */
class Endnote extends Footnote
{
    /**
     * Reference type
     *
     * @var string
     */
    protected $referenceType = 'endnoteReference';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Field.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * Field element writer
 *
 * @since 0.11.0
 */
class Field extends Text
{
    /**
     * Write field element.
     */
    public function write()
    {
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Field) {
            return;
        }

        $methodName = 'write' . ucfirst(strtolower($element->getType()));
        if (method_exists($this, $methodName)) {
            $this->$methodName($element);
        } else {
            $this->writeDefault($element);
        }
    }

    private function writeDefault(\PhpOffice\PhpWord\Element\Field $element)
    {
        $xmlWriter = $this->getXmlWriter();
        $this->startElementP();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $instruction = ' ' . $element->getType() . ' ';
        if ($element->getText() != null) {
            if (is_string($element->getText())) {
                $instruction .= '"' . $element->getText() . '" ';
                $instruction .= $this->buildPropertiesAndOptions($element);
            } else {
                $instruction .= '"';
            }
        } else {
            $instruction .= $this->buildPropertiesAndOptions($element);
        }
        $xmlWriter->startElement('w:r');
        $this->writeFontStyle();
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text($instruction);
        $xmlWriter->endElement(); // w:instrText
        $xmlWriter->endElement(); // w:r

        if ($element->getText() != null) {
            if ($element->getText() instanceof \PhpOffice\PhpWord\Element\TextRun) {
                $containerWriter = new Container($xmlWriter, $element->getText(), true);
                $containerWriter->write();

                $xmlWriter->startElement('w:r');
                $xmlWriter->startElement('w:instrText');
                $xmlWriter->text('"' . $this->buildPropertiesAndOptions($element));
                $xmlWriter->endElement(); // w:instrText
                $xmlWriter->endElement(); // w:r

                $xmlWriter->startElement('w:r');
                $xmlWriter->startElement('w:instrText');
                $xmlWriter->writeAttribute('xml:space', 'preserve');
                $xmlWriter->text(' ');
                $xmlWriter->endElement(); // w:instrText
                $xmlWriter->endElement(); // w:r
            }
        }

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'separate');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:rPr');
        $xmlWriter->startElement('w:noProof');
        $xmlWriter->endElement(); // w:noProof
        $xmlWriter->endElement(); // w:rPr
        $xmlWriter->writeElement('w:t', $element->getText() != null && is_string($element->getText()) ? $element->getText() : '1');
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'end');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }

    /**
     * Writes a macrobutton field
     *
     * //TODO A lot of code duplication with general method, should maybe be refactored
     * @param \PhpOffice\PhpWord\Element\Field $element
     */
    protected function writeMacrobutton(\PhpOffice\PhpWord\Element\Field $element)
    {
        $xmlWriter = $this->getXmlWriter();
        $this->startElementP();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $instruction = ' ' . $element->getType() . ' ' . $this->buildPropertiesAndOptions($element);
        if (is_string($element->getText())) {
            $instruction .= $element->getText() . ' ';
        }

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text($instruction);
        $xmlWriter->endElement(); // w:instrText
        $xmlWriter->endElement(); // w:r

        if ($element->getText() != null) {
            if ($element->getText() instanceof \PhpOffice\PhpWord\Element\TextRun) {
                $containerWriter = new Container($xmlWriter, $element->getText(), true);
                $containerWriter->write();
            }
        }

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'end');
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }

    private function buildPropertiesAndOptions(\PhpOffice\PhpWord\Element\Field $element)
    {
        $propertiesAndOptions = '';
        $properties = $element->getProperties();
        foreach ($properties as $propkey => $propval) {
            switch ($propkey) {
                case 'format':
                    $propertiesAndOptions .= '\* ' . $propval . ' ';
                    break;
                case 'numformat':
                    $propertiesAndOptions .= '\# ' . $propval . ' ';
                    break;
                case 'dateformat':
                    $propertiesAndOptions .= '\@ "' . $propval . '" ';
                    break;
                case 'macroname':
                    $propertiesAndOptions .= $propval . ' ';
                    break;
                default:
                    $propertiesAndOptions .= '"' . $propval . '" ';
                    break;
            }
        }

        $options = $element->getOptions();
        foreach ($options as $option) {
            switch ($option) {
                case 'PreserveFormat':
                    $propertiesAndOptions .= '\* MERGEFORMAT ';
                    break;
                case 'LunarCalendar':
                    $propertiesAndOptions .= '\h ';
                    break;
                case 'SakaEraCalendar':
                    $propertiesAndOptions .= '\s ';
                    break;
                case 'LastUsedFormat':
                    $propertiesAndOptions .= '\l ';
                    break;
                case 'Bold':
                    $propertiesAndOptions .= '\b ';
                    break;
                case 'Italic':
                    $propertiesAndOptions .= '\i ';
                    break;
                default:
                    $propertiesAndOptions .= $option . ' ';
            }
        }

        return $propertiesAndOptions;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Footnote.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * Footnote element writer
 *
 * @since 0.10.0
 */
class Footnote extends Text
{
    /**
     * Reference type footnoteReference|endnoteReference
     *
     * @var string
     */
    protected $referenceType = 'footnoteReference';

    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Footnote) {
            return;
        }

        $this->startElementP();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:rPr');
        $xmlWriter->startElement('w:rStyle');
        $xmlWriter->writeAttribute('w:val', ucfirst($this->referenceType));
        $xmlWriter->endElement(); // w:rStyle
        $xmlWriter->endElement(); // w:rPr
        $xmlWriter->startElement("w:{$this->referenceType}");
        $xmlWriter->writeAttribute('w:id', $element->getRelationId());
        $xmlWriter->endElement(); // w:$referenceType
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/FormField.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\FormField as FormFieldElement;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * FormField element writer
 *
 * Note: DropDown is active when document protection is set to `forms`
 *
 * @since 0.12.0
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_FFData.html
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod)
 */
class FormField extends Text
{
    /** @const int Length of filler when value is null */
    const FILLER_LENGTH = 30;

    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof FormFieldElement) {
            return;
        }

        $type = $element->getType();
        $instructions = array('textinput' => 'FORMTEXT', 'checkbox' => 'FORMCHECKBOX', 'dropdown' => 'FORMDROPDOWN');
        $instruction = $instructions[$type];
        $writeFormField = "write{$type}";
        $name = $element->getName();
        if ($name === null) {
            $name = $type . $element->getElementId();
        }
        $value = $element->getValue();
        if ($value === null) {
            $value = str_repeat(' ', self::FILLER_LENGTH);
        }

        $this->startElementP();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->startElement('w:ffData');
        $xmlWriter->writeElementBlock('w:enabled', 'w:val', 1);
        $xmlWriter->writeElementBlock('w:name', 'w:val', $name);
        $xmlWriter->writeElementBlock('w:calcOnExit', 'w:val', 0);
        $this->$writeFormField($xmlWriter, $element);
        $xmlWriter->endElement(); // w:ffData
        $xmlWriter->endElement(); // w:fldChar
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $this->writeFontStyle();
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text("{$instruction}");
        $xmlWriter->endElement(); // w:instrText
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $this->writeFontStyle();
        $xmlWriter->writeElementBlock('w:fldChar', 'w:fldCharType', 'separate');
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $this->writeFontStyle();
        $xmlWriter->startElement('w:t');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $this->writeText($value);
        $xmlWriter->endElement(); // w:t
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $this->writeFontStyle();
        $xmlWriter->writeElementBlock('w:fldChar', 'w:fldCharType', 'end');
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }

    /**
     * Write textinput.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_FFTextInput.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\FormField $element
     */
    private function writeTextInput(XMLWriter $xmlWriter, FormFieldElement $element)
    {
        $default = $element->getDefault();

        $xmlWriter->startElement('w:textInput');
        $xmlWriter->writeElementBlock('w:default', 'w:val', $default);
        $xmlWriter->endElement();
    }

    /**
     * Write checkbox.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_FFCheckBox.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\FormField $element
     */
    private function writeCheckBox(XMLWriter $xmlWriter, FormFieldElement $element)
    {
        $default = $element->getDefault() ? 1 : 0;
        $value = $element->getValue();
        if ($value == null) {
            $value = $default;
        }
        $value = $value ? 1 : 0;

        $xmlWriter->startElement('w:checkBox');
        $xmlWriter->writeElementBlock('w:sizeAuto', 'w:val', '');
        $xmlWriter->writeElementBlock('w:default', 'w:val', $default);
        $xmlWriter->writeElementBlock('w:checked', 'w:val', $value);
        $xmlWriter->endElement();
    }

    /**
     * Write dropdown.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_FFDDList.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\FormField $element
     */
    private function writeDropDown(XMLWriter $xmlWriter, FormFieldElement $element)
    {
        $default = $element->getDefault();
        $value = $element->getValue();
        if ($value == null) {
            $value = $default;
        }
        $entries = $element->getEntries();

        $xmlWriter->startElement('w:ddList');
        $xmlWriter->writeElementBlock('w:result', 'w:val', $value);
        $xmlWriter->writeElementBlock('w:default', 'w:val', $default);
        foreach ($entries as $entry) {
            if ($entry == null || $entry == '') {
                $entry = str_repeat(' ', self::FILLER_LENGTH);
            }
            $xmlWriter->writeElementBlock('w:listEntry', 'w:val', $entry);
        }
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\Image as ImageElement;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style\Font as FontStyle;
use PhpOffice\PhpWord\Style\Frame as FrameStyle;
use PhpOffice\PhpWord\Writer\Word2007\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Image as ImageStyleWriter;

/**
 * Image element writer
 *
 * @since 0.10.0
 */
class Image extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof ImageElement) {
            return;
        }

        if ($element->isWatermark()) {
            $this->writeWatermark($xmlWriter, $element);
        } else {
            $this->writeImage($xmlWriter, $element);
        }
    }

    /**
     * Write image element.
     */
    private function writeImage(XMLWriter $xmlWriter, ImageElement $element)
    {
        $rId = $element->getRelationId() + ($element->isInSection() ? 6 : 0);
        $style = $element->getStyle();
        $styleWriter = new ImageStyleWriter($xmlWriter, $style);

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
            $styleWriter->writeAlignment();
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');

        // Write position
        $position = $style->getPosition();
        if ($position && $style->getWrap() == FrameStyle::WRAP_INLINE) {
            $fontStyle = new FontStyle('text');
            $fontStyle->setPosition($position);
            $fontStyleWriter = new FontStyleWriter($xmlWriter, $fontStyle);
            $fontStyleWriter->write();
        }

        $xmlWriter->startElement('w:pict');
        $xmlWriter->startElement('v:shape');
        $xmlWriter->writeAttribute('type', '#_x0000_t75');
        $xmlWriter->writeAttribute('stroked', 'f');

        $styleWriter->write();

        $xmlWriter->startElement('v:imagedata');
        $xmlWriter->writeAttribute('r:id', 'rId' . $rId);
        $xmlWriter->writeAttribute('o:title', '');
        $xmlWriter->endElement(); // v:imagedata

        $xmlWriter->endElement(); // v:shape
        $xmlWriter->endElement(); // w:pict
        $xmlWriter->endElement(); // w:r

        $this->endElementP();
    }

    /**
     * Write watermark element.
     */
    private function writeWatermark(XMLWriter $xmlWriter, ImageElement $element)
    {
        $rId = $element->getRelationId();
        $style = $element->getStyle();
        $style->setPositioning('absolute');
        $styleWriter = new ImageStyleWriter($xmlWriter, $style);

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
        }
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:pict');
        $xmlWriter->startElement('v:shape');
        $xmlWriter->writeAttribute('type', '#_x0000_t75');
        $xmlWriter->writeAttribute('stroked', 'f');

        $styleWriter->write();

        $xmlWriter->startElement('v:imagedata');
        $xmlWriter->writeAttribute('r:id', 'rId' . $rId);
        $xmlWriter->writeAttribute('o:title', '');
        $xmlWriter->endElement(); // v:imagedata
        $xmlWriter->endElement(); // v:shape
        $xmlWriter->endElement(); // w:pict
        $xmlWriter->endElement(); // w:r
        if (!$this->withoutP) {
            $xmlWriter->endElement(); // w:p
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Line.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\Line as LineElement;
use PhpOffice\PhpWord\Writer\Word2007\Style\Line as LineStyleWriter;

/**
 * Line element writer
 */
class Line extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof LineElement) {
            return;
        }

        $style = $element->getStyle();
        $styleWriter = new LineStyleWriter($xmlWriter, $style);

        $elementId = $element->getElementIndex();

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
            $styleWriter->writeAlignment();
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:pict');

        // Shapetype could be defined for each line separately, but then a unique id would be necessary
        if ($elementId == 1) {
            $xmlWriter->startElement('v:shapetype');
            $xmlWriter->writeAttribute('id', '_x0000_t32');
            $xmlWriter->writeAttribute('coordsize', '21600,21600');
            $xmlWriter->writeAttribute('o:spt', '32');
            $xmlWriter->writeAttribute('o:oned', 't');
            $xmlWriter->writeAttribute('path', 'm,l21600,21600e');
            $xmlWriter->writeAttribute('filled', 'f');
            $xmlWriter->startElement('v:path');
            $xmlWriter->writeAttribute('arrowok', 't');
            $xmlWriter->writeAttribute('fillok', 'f');
            $xmlWriter->writeAttribute('o:connecttype', 'none');
            $xmlWriter->endElement(); // v:path
            $xmlWriter->startElement('o:lock');
            $xmlWriter->writeAttribute('v:ext', 'edit');
            $xmlWriter->writeAttribute('shapetype', 't');
            $xmlWriter->endElement(); // o:lock
            $xmlWriter->endElement(); // v:shapetype
        }

        $xmlWriter->startElement('v:shape');
        $xmlWriter->writeAttribute('id', sprintf('_x0000_s1%1$03d', $elementId));
        $xmlWriter->writeAttribute('type', '#_x0000_t32'); //type should correspond to shapetype id

        $styleWriter->write();
        $styleWriter->writeStroke();

        $xmlWriter->endElement(); // v:shape

        $xmlWriter->endElement(); // w:pict
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Link.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * Link element writer
 *
 * @since 0.10.0
 */
class Link extends Text
{
    /**
     * Write link element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Link) {
            return;
        }

        $rId = $element->getRelationId() + ($element->isInSection() ? 6 : 0);

        $this->startElementP();

        $xmlWriter->startElement('w:hyperlink');
        if ($element->isInternal()) {
            $xmlWriter->writeAttribute('w:anchor', $element->getSource());
        } else {
            $xmlWriter->writeAttribute('r:id', 'rId' . $rId);
        }
        $xmlWriter->writeAttribute('w:history', '1');
        $xmlWriter->startElement('w:r');

        $this->writeFontStyle();

        $xmlWriter->startElement('w:t');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $this->writeText($element->getText());
        $xmlWriter->endElement(); // w:t
        $xmlWriter->endElement(); // w:r
        $xmlWriter->endElement(); // w:hyperlink

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/ListItem.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Writer\Word2007\Style\Paragraph as ParagraphStyleWriter;

/**
 * ListItem element writer
 *
 * @since 0.10.0
 */
class ListItem extends AbstractElement
{
    /**
     * Write list item element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\ListItem) {
            return;
        }

        $textObject = $element->getTextObject();

        $styleWriter = new ParagraphStyleWriter($xmlWriter, $textObject->getParagraphStyle());
        $styleWriter->setWithoutPPR(true);
        $styleWriter->setIsInline(true);

        $xmlWriter->startElement('w:p');

        $xmlWriter->startElement('w:pPr');
        $styleWriter->write();

        $xmlWriter->startElement('w:numPr');
        $xmlWriter->startElement('w:ilvl');
        $xmlWriter->writeAttribute('w:val', $element->getDepth());
        $xmlWriter->endElement(); // w:ilvl
        $xmlWriter->startElement('w:numId');
        $xmlWriter->writeAttribute('w:val', $element->getStyle()->getNumId());
        $xmlWriter->endElement(); // w:numId
        $xmlWriter->endElement(); // w:numPr

        $xmlWriter->endElement(); // w:pPr

        $elementWriter = new Text($xmlWriter, $textObject, true);
        $elementWriter->write();

        $xmlWriter->endElement(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/ListItemRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\ListItemRun as ListItemRunElement;
use PhpOffice\PhpWord\Writer\Word2007\Style\Paragraph as ParagraphStyleWriter;

/**
 * ListItemRun element writer
 *
 * @since 0.10.0
 */
class ListItemRun extends AbstractElement
{
    /**
     * Write list item element.
     */
    public function write()
    {
        $element = $this->getElement();

        if (!$element instanceof ListItemRunElement) {
            return;
        }

        $this->writeParagraph($element);
    }

    private function writeParagraph(ListItemRunElement $element)
    {
        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('w:p');

        $this->writeParagraphProperties($element);

        $containerWriter = new Container($xmlWriter, $element);
        $containerWriter->write();

        $xmlWriter->endElement(); // w:p
    }

    private function writeParagraphProperties(ListItemRunElement $element)
    {
        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('w:pPr');

        $styleWriter = new ParagraphStyleWriter($xmlWriter, $element->getParagraphStyle());
        $styleWriter->setIsInline(true);
        $styleWriter->setWithoutPPR(true);
        $styleWriter->write();

        $this->writeParagraphPropertiesNumbering($element);

        $xmlWriter->endElement(); // w:pPr
    }

    private function writeParagraphPropertiesNumbering(ListItemRunElement $element)
    {
        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('w:numPr');

        $xmlWriter->writeElementBlock('w:ilvl', array(
            'w:val' => $element->getDepth(),
        ));

        $xmlWriter->writeElementBlock('w:numId', array(
            'w:val' => $element->getStyle()->getNumId(),
        ));

        $xmlWriter->endElement(); // w:numPr
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/OLEObject.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Writer\Word2007\Style\Image as ImageStyleWriter;

/**
 * OLEObject element writer
 *
 * @since 0.10.0
 */
class OLEObject extends AbstractElement
{
    /**
     * Write object element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\OLEObject) {
            return;
        }

        $rIdObject = $element->getRelationId() + ($element->isInSection() ? 6 : 0);
        $rIdImage = $element->getImageRelationId() + ($element->isInSection() ? 6 : 0);
        $shapeId = md5($rIdObject . '_' . $rIdImage);
        $objectId = $element->getRelationId() + 1325353440;

        $style = $element->getStyle();
        $styleWriter = new ImageStyleWriter($xmlWriter, $style);

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
            $styleWriter->writeAlignment();
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:object');
        $xmlWriter->writeAttribute('w:dxaOrig', '249');
        $xmlWriter->writeAttribute('w:dyaOrig', '160');

        // Icon
        $xmlWriter->startElement('v:shape');
        $xmlWriter->writeAttribute('id', $shapeId);
        $xmlWriter->writeAttribute('type', '#_x0000_t75');
        $xmlWriter->writeAttribute('style', 'width:104px;height:67px');
        $xmlWriter->writeAttribute('o:ole', '');

        $xmlWriter->startElement('v:imagedata');
        $xmlWriter->writeAttribute('r:id', 'rId' . $rIdImage);
        $xmlWriter->writeAttribute('o:title', '');
        $xmlWriter->endElement(); // v:imagedata

        $xmlWriter->endElement(); // v:shape

        // Object
        $xmlWriter->startElement('o:OLEObject');
        $xmlWriter->writeAttribute('Type', 'Embed');
        $xmlWriter->writeAttribute('ProgID', 'Package');
        $xmlWriter->writeAttribute('ShapeID', $shapeId);
        $xmlWriter->writeAttribute('DrawAspect', 'Icon');
        $xmlWriter->writeAttribute('ObjectID', '_' . $objectId);
        $xmlWriter->writeAttribute('r:id', 'rId' . $rIdObject);
        $xmlWriter->endElement(); // o:OLEObject

        $xmlWriter->endElement(); // w:object
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/PageBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * PageBreak element writer
 *
 * @since 0.10.0
 */
class PageBreak extends AbstractElement
{
    /**
     * Write element.
     *
     * @usedby \PhpOffice\PhpWord\Writer\Word2007\Element\AbstractElement::startElementP()
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:p');
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:br');
        $xmlWriter->writeAttribute('w:type', 'page');
        $xmlWriter->endElement(); // w:br
        $xmlWriter->endElement(); // w:r
        $xmlWriter->endElement(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/ParagraphAlignment.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * @since 0.13.0
 */
class ParagraphAlignment
{
    private $name = 'w:jc';

    private $attributes = array();

    /**
     * @since 0.13.0
     *
     * @param string $value Any value provided by Jc simple type
     *
     * @see \PhpOffice\PhpWord\SimpleType\Jc For the allowed values of $value parameter.
     */
    final public function __construct($value)
    {
        $this->attributes['w:val'] = $value;
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    final public function getName()
    {
        return $this->name;
    }

    /**
     * @since 0.13.0
     *
     * @return string[]
     */
    final public function getAttributes()
    {
        return $this->attributes;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/PreserveText.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * PreserveText element writer
 *
 * @since 0.10.0
 */
class PreserveText extends Text
{
    /**
     * Write preserve text element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\PreserveText) {
            return;
        }

        $texts = $element->getText();
        if (!is_array($texts)) {
            $texts = array($texts);
        }

        $this->startElementP();

        foreach ($texts as $text) {
            if (substr($text, 0, 1) == '{') {
                $text = substr($text, 1, -1);

                $xmlWriter->startElement('w:r');
                $xmlWriter->startElement('w:fldChar');
                $xmlWriter->writeAttribute('w:fldCharType', 'begin');
                $xmlWriter->endElement();
                $xmlWriter->endElement();

                $xmlWriter->startElement('w:r');

                $this->writeFontStyle();

                $xmlWriter->startElement('w:instrText');
                $xmlWriter->writeAttribute('xml:space', 'preserve');
                $this->writeText($text);
                $xmlWriter->endElement();
                $xmlWriter->endElement();

                $xmlWriter->startElement('w:r');
                $xmlWriter->startElement('w:fldChar');
                $xmlWriter->writeAttribute('w:fldCharType', 'separate');
                $xmlWriter->endElement();
                $xmlWriter->endElement();

                $xmlWriter->startElement('w:r');
                $xmlWriter->startElement('w:fldChar');
                $xmlWriter->writeAttribute('w:fldCharType', 'end');
                $xmlWriter->endElement();
                $xmlWriter->endElement();
            } else {
                $xmlWriter->startElement('w:r');

                $this->writeFontStyle();

                $xmlWriter->startElement('w:t');
                $xmlWriter->writeAttribute('xml:space', 'preserve');
                $this->writeText($this->getText($text));
                $xmlWriter->endElement();
                $xmlWriter->endElement();
            }
        }

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/SDT.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\SDT as SDTElement;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Structured document tag element writer
 *
 * @since 0.12.0
 * @see  http://www.datypic.com/sc/ooxml/t-w_CT_SdtBlock.html
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod)
 */
class SDT extends Text
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof SDTElement) {
            return;
        }
        $type = $element->getType();
        $writeFormField = "write{$type}";
        $alias = $element->getAlias();
        $tag = $element->getTag();
        $value = $element->getValue();
        if ($value === null) {
            $value = 'Pick value';
        }

        $this->startElementP();

        $xmlWriter->startElement('w:sdt');

        // Properties
        $xmlWriter->startElement('w:sdtPr');
        $xmlWriter->writeElementIf($alias != null, 'w:alias', 'w:val', $alias);
        $xmlWriter->writeElementBlock('w:lock', 'w:val', 'sdtLocked');
        $xmlWriter->writeElementBlock('w:id', 'w:val', rand(100000000, 999999999));
        $xmlWriter->writeElementIf($tag != null, 'w:tag', 'w:val', $tag);
        $this->$writeFormField($xmlWriter, $element);
        $xmlWriter->endElement(); // w:sdtPr

        // Content
        $xmlWriter->startElement('w:sdtContent');
        $xmlWriter->startElement('w:r');
        $xmlWriter->writeElement('w:t', $value);
        $xmlWriter->endElement(); // w:r
        $xmlWriter->endElement(); // w:sdtContent

        $xmlWriter->endElement(); // w:sdt

        $this->endElementP(); // w:p
    }

    /**
     * Write text.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_SdtText.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writePlainText(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('w:text');
        $xmlWriter->endElement(); // w:text
    }

    /**
     * Write combo box.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_SdtComboBox.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\SDT $element
     */
    private function writeComboBox(XMLWriter $xmlWriter, SDTElement $element)
    {
        $type = $element->getType();
        $listItems = $element->getListItems();

        $xmlWriter->startElement("w:{$type}");
        foreach ($listItems as $key => $val) {
            $xmlWriter->writeElementBlock('w:listItem', array('w:value' => $key, 'w:displayText' => $val));
        }
        $xmlWriter->endElement(); // w:{$type}
    }

    /**
     * Write drop down list.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_SdtDropDownList.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\SDT $element
     */
    private function writeDropDownList(XMLWriter $xmlWriter, SDTElement $element)
    {
        $this->writeComboBox($xmlWriter, $element);
    }

    /**
     * Write date.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-w_CT_SdtDate.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\SDT $element
     */
    private function writeDate(XMLWriter $xmlWriter, SDTElement $element)
    {
        $type = $element->getType();

        $xmlWriter->startElement("w:{$type}");
        $xmlWriter->writeElementBlock('w:dateFormat', 'w:val', 'd/M/yyyy');
        $xmlWriter->writeElementBlock('w:lid', 'w:val', 'en-US');
        $xmlWriter->writeElementBlock('w:storeMappedDataAs', 'w:val', 'dateTime');
        $xmlWriter->writeElementBlock('w:calendar', 'w:val', 'gregorian');
        $xmlWriter->endElement(); // w:date
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Shape.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\Shape as ShapeElement;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style\Shape as ShapeStyle;
use PhpOffice\PhpWord\Writer\Word2007\Style\Shape as ShapeStyleWriter;

/**
 * Shape element writer
 *
 * @since 0.12.0
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod)
 */
class Shape extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof ShapeElement) {
            return;
        }

        $style = $element->getStyle();
        $styleWriter = new ShapeStyleWriter($xmlWriter, $style);

        $type = $element->getType();
        if ($type == 'rect' && $style->getRoundness() !== null) {
            $type = 'roundrect';
        }
        $method = "write{$type}";

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:pict');
        $xmlWriter->startElement("v:{$type}");

        // Element style
        if (method_exists($this, $method)) {
            $this->$method($xmlWriter, $style);
        }

        // Child style
        $styleWriter->write();

        $xmlWriter->endElement(); // v:$type
        $xmlWriter->endElement(); // w:pict
        $xmlWriter->endElement(); // w:r

        $this->endElementP(); // w:p
    }

    /**
     * Write arc.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Shape $style
     */
    private function writeArc(XMLWriter $xmlWriter, ShapeStyle $style)
    {
        $points = $this->getPoints('arc', $style->getPoints());

        $xmlWriter->writeAttributeIf($points['start'] !== null, 'startAngle', $points['start']);
        $xmlWriter->writeAttributeIf($points['end'] !== null, 'endAngle', $points['end']);
    }

    /**
     * Write curve.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Shape $style
     */
    private function writeCurve(XMLWriter $xmlWriter, ShapeStyle $style)
    {
        $points = $this->getPoints('curve', $style->getPoints());

        $this->writeLine($xmlWriter, $style);
        $xmlWriter->writeAttributeIf($points['point1'] !== null, 'control1', $points['point1']);
        $xmlWriter->writeAttributeIf($points['point2'] !== null, 'control2', $points['point2']);
    }

    /**
     * Write line.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Shape $style
     */
    private function writeLine(XMLWriter $xmlWriter, ShapeStyle $style)
    {
        $points = $this->getPoints('line', $style->getPoints());

        $xmlWriter->writeAttributeIf($points['start'] !== null, 'from', $points['start']);
        $xmlWriter->writeAttributeIf($points['end'] !== null, 'to', $points['end']);
    }

    /**
     * Write polyline.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Shape $style
     */
    private function writePolyline(XMLWriter $xmlWriter, ShapeStyle $style)
    {
        $xmlWriter->writeAttributeIf($style->getPoints() !== null, 'points', $style->getPoints());
    }

    /**
     * Write rectangle.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Shape $style
     */
    private function writeRoundRect(XMLWriter $xmlWriter, ShapeStyle $style)
    {
        $xmlWriter->writeAttribute('arcsize', $style->getRoundness());
    }

    /**
     * Set points
     *
     * @param string $type
     * @param string $value
     * @return array
     */
    private function getPoints($type, $value)
    {
        $points = array();

        switch ($type) {
            case 'arc':
            case 'line':
                $points = explode(' ', $value);
                list($start, $end) = array_pad($points, 2, null);
                $points = array('start' => $start, 'end' => $end);
                break;
            case 'curve':
                $points = explode(' ', $value);
                list($start, $end, $point1, $point2) = array_pad($points, 4, null);
                $points = array('start' => $start, 'end' => $end, 'point1' => $point1, 'point2' => $point2);
                break;
        }

        return $points;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\Cell as CellElement;
use PhpOffice\PhpWord\Element\Row as RowElement;
use PhpOffice\PhpWord\Element\Table as TableElement;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style\Cell as CellStyle;
use PhpOffice\PhpWord\Style\Row as RowStyle;
use PhpOffice\PhpWord\Writer\Word2007\Style\Cell as CellStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Row as RowStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Table as TableStyleWriter;

/**
 * Table element writer
 *
 * @since 0.10.0
 */
class Table extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof TableElement) {
            return;
        }

        $rows = $element->getRows();
        $rowCount = count($rows);

        if ($rowCount > 0) {
            $xmlWriter->startElement('w:tbl');

            // Write columns
            $this->writeColumns($xmlWriter, $element);

            // Write style
            $styleWriter = new TableStyleWriter($xmlWriter, $element->getStyle());
            $styleWriter->setWidth($element->getWidth());
            $styleWriter->write();

            // Write rows
            for ($i = 0; $i < $rowCount; $i++) {
                $this->writeRow($xmlWriter, $rows[$i]);
            }

            $xmlWriter->endElement(); // w:tbl
        }
    }

    /**
     * Write column.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Table $element
     */
    private function writeColumns(XMLWriter $xmlWriter, TableElement $element)
    {
        $cellWidths = $element->findFirstDefinedCellWidths();

        $xmlWriter->startElement('w:tblGrid');
        foreach ($cellWidths as $width) {
            $xmlWriter->startElement('w:gridCol');
            if ($width !== null) {
                $xmlWriter->writeAttribute('w:w', $width);
                $xmlWriter->writeAttribute('w:type', 'dxa');
            }
            $xmlWriter->endElement();
        }
        $xmlWriter->endElement(); // w:tblGrid
    }

    /**
     * Write row.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Row $row
     */
    private function writeRow(XMLWriter $xmlWriter, RowElement $row)
    {
        $xmlWriter->startElement('w:tr');

        // Write style
        $rowStyle = $row->getStyle();
        if ($rowStyle instanceof RowStyle) {
            $styleWriter = new RowStyleWriter($xmlWriter, $rowStyle);
            $styleWriter->setHeight($row->getHeight());
            $styleWriter->write();
        }

        // Write cells
        foreach ($row->getCells() as $cell) {
            $this->writeCell($xmlWriter, $cell);
        }

        $xmlWriter->endElement(); // w:tr
    }

    /**
     * Write cell.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Cell $cell
     */
    private function writeCell(XMLWriter $xmlWriter, CellElement $cell)
    {
        $xmlWriter->startElement('w:tc');

        // Write style
        $cellStyle = $cell->getStyle();
        if ($cellStyle instanceof CellStyle) {
            $styleWriter = new CellStyleWriter($xmlWriter, $cellStyle);
            $styleWriter->setWidth($cell->getWidth());
            $styleWriter->write();
        }

        // Write content
        $containerWriter = new Container($xmlWriter, $cell);
        $containerWriter->write();

        $xmlWriter->endElement(); // w:tc
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/TableAlignment.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * @since 0.13.0
 */
class TableAlignment
{
    private $name = 'w:jc';

    private $attributes = array();

    /**
     * @since 0.13.0
     *
     * @param string $value Any value provided by JcTable simple type
     *
     * @see \PhpOffice\PhpWord\SimpleType\JcTable For the allowed values of $value parameter.
     */
    final public function __construct($value)
    {
        $this->attributes['w:val'] = $value;
    }

    /**
     * @since 0.13.0
     *
     * @return string
     */
    final public function getName()
    {
        return $this->name;
    }

    /**
     * @since 0.13.0
     *
     * @return string[]
     */
    final public function getAttributes()
    {
        return $this->attributes;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Text.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\TrackChange;

/**
 * Text element writer
 *
 * @since 0.10.0
 */
class Text extends AbstractElement
{
    /**
     * Write text element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Text) {
            return;
        }

        $this->startElementP();

        $this->writeOpeningTrackChange();

        $xmlWriter->startElement('w:r');

        $this->writeFontStyle();

        $textElement = 'w:t';
        //'w:delText' in case of deleted text
        $changed = $element->getTrackChange();
        if ($changed != null && $changed->getChangeType() == TrackChange::DELETED) {
            $textElement = 'w:delText';
        }
        $xmlWriter->startElement($textElement);

        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $this->writeText($this->getText($element->getText()));
        $xmlWriter->endElement();
        $xmlWriter->endElement(); // w:r

        $this->writeClosingTrackChange();

        $this->endElementP(); // w:p
    }

    /**
     * Write opening of changed element
     */
    protected function writeOpeningTrackChange()
    {
        $changed = $this->getElement()->getTrackChange();
        if ($changed == null) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();

        if (($changed->getChangeType() == TrackChange::INSERTED)) {
            $xmlWriter->startElement('w:ins');
        } elseif ($changed->getChangeType() == TrackChange::DELETED) {
            $xmlWriter->startElement('w:del');
        }
        $xmlWriter->writeAttribute('w:author', $changed->getAuthor());
        if ($changed->getDate() != null) {
            $xmlWriter->writeAttribute('w:date', $changed->getDate()->format('Y-m-d\TH:i:s\Z'));
        }
        $xmlWriter->writeAttribute('w:id', $this->getElement()->getElementId());
    }

    /**
     * Write ending
     */
    protected function writeClosingTrackChange()
    {
        $changed = $this->getElement()->getTrackChange();
        if ($changed == null) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->endElement(); // w:ins|w:del
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/TextBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Writer\Word2007\Style\TextBox as TextBoxStyleWriter;

/**
 * TextBox element writer
 *
 * @since 0.11.0
 */
class TextBox extends Image
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\TextBox) {
            return;
        }
        $style = $element->getStyle();
        $styleWriter = new TextBoxStyleWriter($xmlWriter, $style);

        if (!$this->withoutP) {
            $xmlWriter->startElement('w:p');
            $styleWriter->writeAlignment();
        }
        $this->writeCommentRangeStart();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:pict');
        $xmlWriter->startElement('v:shape');
        $xmlWriter->writeAttribute('type', '#_x0000_t0202');

        $styleWriter->write();
        $styleWriter->writeBorder();

        $xmlWriter->startElement('v:textbox');
        $styleWriter->writeInnerMargin();

        // TextBox content, serving as a container
        $xmlWriter->startElement('w:txbxContent');
        $containerWriter = new Container($xmlWriter, $element);
        $containerWriter->write();
        $xmlWriter->endElement(); // w:txbxContent

        $xmlWriter->endElement(); // v: textbox

        $xmlWriter->endElement(); // v:shape
        $xmlWriter->endElement(); // w:pict
        $xmlWriter->endElement(); // w:r

        $this->endElementP();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/TextBreak.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * TextBreak element writer
 *
 * @since 0.10.0
 */
class TextBreak extends Text
{
    /**
     * Write text break element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\TextBreak) {
            return;
        }

        if (!$this->withoutP) {
            $hasStyle = $element->hasStyle();
            $this->startElementP();

            if ($hasStyle) {
                $xmlWriter->startElement('w:pPr');
                $this->writeFontStyle();
                $xmlWriter->endElement(); // w:pPr
            }

            $this->endElementP(); // w:p
        } else {
            $xmlWriter->writeElement('w:br');
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/TextRun.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * TextRun element writer
 *
 * @since 0.10.0
 */
class TextRun extends Text
{
    /**
     * Write textrun element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();

        $this->startElementP();

        $containerWriter = new Container($xmlWriter, $element);
        $containerWriter->write();

        $this->endElementP(); // w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/Title.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

/**
 * TextRun element writer
 *
 * @since 0.10.0
 */
class Title extends AbstractElement
{
    /**
     * Write title element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof \PhpOffice\PhpWord\Element\Title) {
            return;
        }

        $style = $element->getStyle();

        $xmlWriter->startElement('w:p');

        if (!empty($style)) {
            $xmlWriter->startElement('w:pPr');
            $xmlWriter->startElement('w:pStyle');
            $xmlWriter->writeAttribute('w:val', $style);
            $xmlWriter->endElement();
            $xmlWriter->endElement();
        }

        $bookmarkRId = null;
        if ($element->getDepth() !== 0) {
            $rId = $element->getRelationId();
            $bookmarkRId = $element->getPhpWord()->addBookmark();

            // Bookmark start for TOC
            $xmlWriter->startElement('w:bookmarkStart');
            $xmlWriter->writeAttribute('w:id', $bookmarkRId);
            $xmlWriter->writeAttribute('w:name', "_Toc{$rId}");
            $xmlWriter->endElement(); //w:bookmarkStart
        }

        // Actual text
        $text = $element->getText();
        if (is_string($text)) {
            $xmlWriter->startElement('w:r');
            $xmlWriter->startElement('w:t');
            $this->writeText($text);
            $xmlWriter->endElement(); // w:t
            $xmlWriter->endElement(); // w:r
        } elseif ($text instanceof \PhpOffice\PhpWord\Element\AbstractContainer) {
            $containerWriter = new Container($xmlWriter, $text);
            $containerWriter->write();
        }

        if ($element->getDepth() !== 0) {
            // Bookmark end
            $xmlWriter->startElement('w:bookmarkEnd');
            $xmlWriter->writeAttribute('w:id', $bookmarkRId);
            $xmlWriter->endElement(); //w:bookmarkEnd
        }
        $xmlWriter->endElement(); //w:p
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Element/TOC.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Element;

use PhpOffice\PhpWord\Element\TOC as TOCElement;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style\Font;
use PhpOffice\PhpWord\Writer\Word2007\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Paragraph as ParagraphStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Tab as TabStyleWriter;

/**
 * TOC element writer
 *
 * @since 0.10.0
 */
class TOC extends AbstractElement
{
    /**
     * Write element.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $element = $this->getElement();
        if (!$element instanceof TOCElement) {
            return;
        }

        $titles = $element->getTitles();
        $writeFieldMark = true;

        foreach ($titles as $title) {
            $this->writeTitle($xmlWriter, $element, $title, $writeFieldMark);
            if ($writeFieldMark) {
                $writeFieldMark = false;
            }
        }

        $xmlWriter->startElement('w:p');
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'end');
        $xmlWriter->endElement();
        $xmlWriter->endElement();
        $xmlWriter->endElement();
    }

    /**
     * Write title
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\TOC $element
     * @param \PhpOffice\PhpWord\Element\Title $title
     * @param bool $writeFieldMark
     */
    private function writeTitle(XMLWriter $xmlWriter, TOCElement $element, $title, $writeFieldMark)
    {
        $tocStyle = $element->getStyleTOC();
        $fontStyle = $element->getStyleFont();
        $isObject = ($fontStyle instanceof Font) ? true : false;
        $rId = $title->getRelationId();
        $indent = ($title->getDepth() - 1) * $tocStyle->getIndent();

        $xmlWriter->startElement('w:p');

        // Write style and field mark
        $this->writeStyle($xmlWriter, $element, $indent);
        if ($writeFieldMark) {
            $this->writeFieldMark($xmlWriter, $element);
        }

        // Hyperlink
        $xmlWriter->startElement('w:hyperlink');
        $xmlWriter->writeAttribute('w:anchor', "_Toc{$rId}");
        $xmlWriter->writeAttribute('w:history', '1');

        // Title text
        $xmlWriter->startElement('w:r');
        if ($isObject) {
            $styleWriter = new FontStyleWriter($xmlWriter, $fontStyle);
            $styleWriter->write();
        }
        $xmlWriter->startElement('w:t');
        $this->writeText($title->getText());
        $xmlWriter->endElement(); // w:t
        $xmlWriter->endElement(); // w:r

        $xmlWriter->startElement('w:r');
        $xmlWriter->writeElement('w:tab', null);
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->endElement();
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text("PAGEREF _Toc{$rId} \h");
        $xmlWriter->endElement();
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'end');
        $xmlWriter->endElement();
        $xmlWriter->endElement();

        $xmlWriter->endElement(); // w:hyperlink

        $xmlWriter->endElement(); // w:p
    }

    /**
     * Write style
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\TOC $element
     * @param int $indent
     */
    private function writeStyle(XMLWriter $xmlWriter, TOCElement $element, $indent)
    {
        $tocStyle = $element->getStyleTOC();
        $fontStyle = $element->getStyleFont();
        $isObject = ($fontStyle instanceof Font) ? true : false;

        $xmlWriter->startElement('w:pPr');

        // Paragraph
        if ($isObject && !is_null($fontStyle->getParagraph())) {
            $styleWriter = new ParagraphStyleWriter($xmlWriter, $fontStyle->getParagraph());
            $styleWriter->write();
        }

        // Font
        if (!empty($fontStyle) && !$isObject) {
            $xmlWriter->startElement('w:rPr');
            $xmlWriter->startElement('w:rStyle');
            $xmlWriter->writeAttribute('w:val', $fontStyle);
            $xmlWriter->endElement();
            $xmlWriter->endElement(); // w:rPr
        }

        // Tab
        $xmlWriter->startElement('w:tabs');
        $styleWriter = new TabStyleWriter($xmlWriter, $tocStyle);
        $styleWriter->write();
        $xmlWriter->endElement();

        // Indent
        if ($indent > 0) {
            $xmlWriter->startElement('w:ind');
            $xmlWriter->writeAttribute('w:left', $indent);
            $xmlWriter->endElement();
        }

        $xmlWriter->endElement(); // w:pPr
    }

    /**
     * Write TOC Field.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\TOC $element
     */
    private function writeFieldMark(XMLWriter $xmlWriter, TOCElement $element)
    {
        $minDepth = $element->getMinDepth();
        $maxDepth = $element->getMaxDepth();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'begin');
        $xmlWriter->endElement();
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:instrText');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text("TOC \o {$minDepth}-{$maxDepth} \h \z \u");
        $xmlWriter->endElement();
        $xmlWriter->endElement();

        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:fldChar');
        $xmlWriter->writeAttribute('w:fldCharType', 'separate');
        $xmlWriter->endElement();
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/AbstractPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Writer\AbstractWriter;

/**
 * Word2007 writer part abstract class
 */
abstract class AbstractPart
{
    /**
     * Parent writer
     *
     * @var \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    protected $parentWriter;

    /**
     * @var string Date format
     */
    protected $dateFormat = 'Y-m-d\TH:i:sP';

    /**
     * Write part
     *
     * @return string
     */
    abstract public function write();

    /**
     * Set parent writer.
     *
     * @param \PhpOffice\PhpWord\Writer\AbstractWriter $writer
     */
    public function setParentWriter(AbstractWriter $writer = null)
    {
        $this->parentWriter = $writer;
    }

    /**
     * Get parent writer
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     * @return \PhpOffice\PhpWord\Writer\AbstractWriter
     */
    public function getParentWriter()
    {
        if (!is_null($this->parentWriter)) {
            return $this->parentWriter;
        }
        throw new Exception('No parent WriterInterface assigned.');
    }

    /**
     * Get XML Writer
     *
     * @return \PhpOffice\PhpWord\Shared\XMLWriter
     */
    protected function getXmlWriter()
    {
        $useDiskCaching = false;
        if (!is_null($this->parentWriter)) {
            if ($this->parentWriter->isUseDiskCaching()) {
                $useDiskCaching = true;
            }
        }
        if ($useDiskCaching) {
            return new XMLWriter(XMLWriter::STORAGE_DISK, $this->parentWriter->getDiskCachingDirectory(), Settings::hasCompatibility());
        }

        return new XMLWriter(XMLWriter::STORAGE_MEMORY, './', Settings::hasCompatibility());
    }

    /**
     * Write an XML text, this will call text() or writeRaw() depending on the value of Settings::isOutputEscapingEnabled()
     *
     * @param string $content The text string to write
     * @return bool Returns true on success or false on failure
     */
    protected function writeText($content)
    {
        if (Settings::isOutputEscapingEnabled()) {
            return $this->getXmlWriter()->text($content);
        }

        return $this->getXmlWriter()->writeRaw($content);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Chart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Element\Chart as ChartElement;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Word2007 chart part writer: word/charts/chartx.xml
 *
 * @since 0.12.0
 * @see  http://www.datypic.com/sc/ooxml/e-draw-chart_chartSpace.html
 */
class Chart extends AbstractPart
{
    /**
     * Chart element
     *
     * @var \PhpOffice\PhpWord\Element\Chart
     */
    private $element;

    /**
     * Type definition
     *
     * @var array
     */
    private $types = array(
        'pie'                    => array('type' => 'pie', 'colors' => 1),
        'doughnut'               => array('type' => 'doughnut', 'colors' => 1, 'hole' => 75, 'no3d' => true),
        'bar'                    => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'bar', 'grouping' => 'clustered'),
        'stacked_bar'            => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'bar', 'grouping' => 'stacked'),
        'percent_stacked_bar'    => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'bar', 'grouping' => 'percentStacked'),
        'column'                 => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'col', 'grouping' => 'clustered'),
        'stacked_column'         => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'col', 'grouping' => 'stacked'),
        'percent_stacked_column' => array('type' => 'bar', 'colors' => 0, 'axes' => true, 'bar' => 'col', 'grouping' => 'percentStacked'),
        'line'                   => array('type' => 'line', 'colors' => 0, 'axes' => true),
        'area'                   => array('type' => 'area', 'colors' => 0, 'axes' => true),
        'radar'                  => array('type' => 'radar', 'colors' => 0, 'axes' => true, 'radar' => 'standard', 'no3d' => true),
        'scatter'                => array('type' => 'scatter', 'colors' => 0, 'axes' => true, 'scatter' => 'marker', 'no3d' => true),
    );

    /**
     * Chart options
     *
     * @var array
     */
    private $options = array();

    /**
     * Set chart element.
     *
     * @param \PhpOffice\PhpWord\Element\Chart $element
     */
    public function setElement(ChartElement $element)
    {
        $this->element = $element;
    }

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('c:chartSpace');
        $xmlWriter->writeAttribute('xmlns:c', 'http://schemas.openxmlformats.org/drawingml/2006/chart');
        $xmlWriter->writeAttribute('xmlns:a', 'http://schemas.openxmlformats.org/drawingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');

        $this->writeChart($xmlWriter);
        $this->writeShape($xmlWriter);

        $xmlWriter->endElement(); // c:chartSpace

        return $xmlWriter->getData();
    }

    /**
     * Write chart
     *
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_Chart.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writeChart(XMLWriter $xmlWriter)
    {
        $xmlWriter->startElement('c:chart');

        $this->writePlotArea($xmlWriter);

        $xmlWriter->endElement(); // c:chart
    }

    /**
     * Write plot area.
     *
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_PlotArea.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_PieChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_DoughnutChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_BarChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_LineChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_AreaChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_RadarChart.html
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_ScatterChart.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     */
    private function writePlotArea(XMLWriter $xmlWriter)
    {
        $type = $this->element->getType();
        $style = $this->element->getStyle();
        $this->options = $this->types[$type];

        $title = $style->getTitle();
        $showLegend = $style->isShowLegend();
        $legendPosition = $style->getLegendPosition();

        //Chart title
        if ($title) {
            $xmlWriter->startElement('c:title');
            $xmlWriter->startElement('c:tx');
            $xmlWriter->startElement('c:rich');
            $xmlWriter->writeRaw('
                <a:bodyPr/>
                <a:lstStyle/>
                <a:p>
                <a:pPr>
                <a:defRPr/></a:pPr><a:r><a:rPr/><a:t>' . $title . '</a:t></a:r>
                <a:endParaRPr/>
                </a:p>');
            $xmlWriter->endElement(); // c:rich
            $xmlWriter->endElement(); // c:tx
            $xmlWriter->endElement(); // c:title
        } else {
            $xmlWriter->writeElementBlock('c:autoTitleDeleted', 'val', 1);
        }

        //Chart legend
        if ($showLegend) {
            $xmlWriter->writeRaw('<c:legend><c:legendPos val="' . $legendPosition . '"/></c:legend>');
        }

        $xmlWriter->startElement('c:plotArea');
        $xmlWriter->writeElement('c:layout');

        // Chart
        $chartType = $this->options['type'];
        $chartType .= $style->is3d() && !isset($this->options['no3d']) ? '3D' : '';
        $chartType .= 'Chart';
        $xmlWriter->startElement("c:{$chartType}");

        $xmlWriter->writeElementBlock('c:varyColors', 'val', $this->options['colors']);
        if ($type == 'area') {
            $xmlWriter->writeElementBlock('c:grouping', 'val', 'standard');
        }
        if (isset($this->options['hole'])) {
            $xmlWriter->writeElementBlock('c:holeSize', 'val', $this->options['hole']);
        }
        if (isset($this->options['bar'])) {
            $xmlWriter->writeElementBlock('c:barDir', 'val', $this->options['bar']); // bar|col
            $xmlWriter->writeElementBlock('c:grouping', 'val', $this->options['grouping']); // 3d; standard = percentStacked
        }
        if (isset($this->options['radar'])) {
            $xmlWriter->writeElementBlock('c:radarStyle', 'val', $this->options['radar']);
        }
        if (isset($this->options['scatter'])) {
            $xmlWriter->writeElementBlock('c:scatterStyle', 'val', $this->options['scatter']);
        }

        // Series
        $this->writeSeries($xmlWriter, isset($this->options['scatter']));

        // don't overlap if grouping is 'clustered'
        if (!isset($this->options['grouping']) || $this->options['grouping'] != 'clustered') {
            $xmlWriter->writeElementBlock('c:overlap', 'val', '100');
        }

        // Axes
        if (isset($this->options['axes'])) {
            $xmlWriter->writeElementBlock('c:axId', 'val', 1);
            $xmlWriter->writeElementBlock('c:axId', 'val', 2);
        }

        $xmlWriter->endElement(); // chart type

        // Axes
        if (isset($this->options['axes'])) {
            $this->writeAxis($xmlWriter, 'cat');
            $this->writeAxis($xmlWriter, 'val');
        }

        $xmlWriter->endElement(); // c:plotArea
    }

    /**
     * Write series.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param bool $scatter
     */
    private function writeSeries(XMLWriter $xmlWriter, $scatter = false)
    {
        $series = $this->element->getSeries();
        $style = $this->element->getStyle();
        $colors = $style->getColors();

        $index = 0;
        $colorIndex = 0;
        foreach ($series as $seriesItem) {
            $categories = $seriesItem['categories'];
            $values = $seriesItem['values'];

            $xmlWriter->startElement('c:ser');

            $xmlWriter->writeElementBlock('c:idx', 'val', $index);
            $xmlWriter->writeElementBlock('c:order', 'val', $index);

            if (!is_null($seriesItem['name']) && $seriesItem['name'] != '') {
                $xmlWriter->startElement('c:tx');
                $xmlWriter->startElement('c:strRef');
                $xmlWriter->startElement('c:strCache');
                $xmlWriter->writeElementBlock('c:ptCount', 'val', 1);
                $xmlWriter->startElement('c:pt');
                $xmlWriter->writeAttribute('idx', 0);
                $xmlWriter->startElement('c:v');
                $xmlWriter->writeRaw($seriesItem['name']);
                $xmlWriter->endElement(); // c:v
                $xmlWriter->endElement(); // c:pt
                $xmlWriter->endElement(); // c:strCache
                $xmlWriter->endElement(); // c:strRef
                $xmlWriter->endElement(); // c:tx
            }

            // The c:dLbls was added to make word charts look more like the reports in SurveyGizmo
            // This section needs to be made configurable before a pull request is made
            $xmlWriter->startElement('c:dLbls');

            foreach ($style->getDataLabelOptions() as $option => $val) {
                $xmlWriter->writeElementBlock("c:{$option}", 'val', (int) $val);
            }

            $xmlWriter->endElement(); // c:dLbls

            if (isset($this->options['scatter'])) {
                $this->writeShape($xmlWriter);
            }

            if ($scatter === true) {
                $this->writeSeriesItem($xmlWriter, 'xVal', $categories);
                $this->writeSeriesItem($xmlWriter, 'yVal', $values);
            } else {
                $this->writeSeriesItem($xmlWriter, 'cat', $categories);
                $this->writeSeriesItem($xmlWriter, 'val', $values);

                // check that there are colors
                if (is_array($colors) && count($colors) > 0) {
                    // assign a color to each value
                    $valueIndex = 0;
                    for ($i = 0; $i < count($values); $i++) {
                        // check that there are still enought colors
                        $xmlWriter->startElement('c:dPt');
                        $xmlWriter->writeElementBlock('c:idx', 'val', $valueIndex);
                        $xmlWriter->startElement('c:spPr');
                        $xmlWriter->startElement('a:solidFill');
                        $xmlWriter->writeElementBlock('a:srgbClr', 'val', $colors[$colorIndex++ % count($colors)]);
                        $xmlWriter->endElement(); // a:solidFill
                        $xmlWriter->endElement(); // c:spPr
                        $xmlWriter->endElement(); // c:dPt
                        $valueIndex++;
                    }
                }
            }

            $xmlWriter->endElement(); // c:ser
            $index++;
        }
    }

    /**
     * Write series items.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $type
     * @param array $values
     */
    private function writeSeriesItem(XMLWriter $xmlWriter, $type, $values)
    {
        $types = array(
            'cat'  => array('c:cat', 'c:strLit'),
            'val'  => array('c:val', 'c:numLit'),
            'xVal' => array('c:xVal', 'c:strLit'),
            'yVal' => array('c:yVal', 'c:numLit'),
        );
        list($itemType, $itemLit) = $types[$type];

        $xmlWriter->startElement($itemType);
        $xmlWriter->startElement($itemLit);
        $xmlWriter->writeElementBlock('c:ptCount', 'val', count($values));

        $index = 0;
        foreach ($values as $value) {
            $xmlWriter->startElement('c:pt');
            $xmlWriter->writeAttribute('idx', $index);
            if (\PhpOffice\PhpWord\Settings::isOutputEscapingEnabled()) {
                $xmlWriter->writeElement('c:v', $value);
            } else {
                $xmlWriter->startElement('c:v');
                $xmlWriter->writeRaw($value);
                $xmlWriter->endElement(); // c:v
            }
            $xmlWriter->endElement(); // c:pt
            $index++;
        }

        $xmlWriter->endElement(); // $itemLit
        $xmlWriter->endElement(); // $itemType
    }

    /**
     * Write axis
     *
     * @see  http://www.datypic.com/sc/ooxml/t-draw-chart_CT_CatAx.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $type
     */
    private function writeAxis(XMLWriter $xmlWriter, $type)
    {
        $style = $this->element->getStyle();
        $types = array(
            'cat' => array('c:catAx', 1, 'b', 2),
            'val' => array('c:valAx', 2, 'l', 1),
        );
        list($axisType, $axisId, $axisPos, $axisCross) = $types[$type];

        $xmlWriter->startElement($axisType);

        $xmlWriter->writeElementBlock('c:axId', 'val', $axisId);
        $xmlWriter->writeElementBlock('c:axPos', 'val', $axisPos);

        $categoryAxisTitle = $style->getCategoryAxisTitle();
        $valueAxisTitle = $style->getValueAxisTitle();

        if ($axisType == 'c:catAx') {
            if (!is_null($categoryAxisTitle)) {
                $this->writeAxisTitle($xmlWriter, $categoryAxisTitle);
            }
        } elseif ($axisType == 'c:valAx') {
            if (!is_null($valueAxisTitle)) {
                $this->writeAxisTitle($xmlWriter, $valueAxisTitle);
            }
        }

        $xmlWriter->writeElementBlock('c:crossAx', 'val', $axisCross);
        $xmlWriter->writeElementBlock('c:auto', 'val', 1);

        if (isset($this->options['axes'])) {
            $xmlWriter->writeElementBlock('c:delete', 'val', 0);
            $xmlWriter->writeElementBlock('c:majorTickMark', 'val', $style->getMajorTickPosition());
            $xmlWriter->writeElementBlock('c:minorTickMark', 'val', 'none');
            if ($style->showAxisLabels()) {
                if ($axisType == 'c:catAx') {
                    $xmlWriter->writeElementBlock('c:tickLblPos', 'val', $style->getCategoryLabelPosition());
                } else {
                    $xmlWriter->writeElementBlock('c:tickLblPos', 'val', $style->getValueLabelPosition());
                }
            } else {
                $xmlWriter->writeElementBlock('c:tickLblPos', 'val', 'none');
            }
            $xmlWriter->writeElementBlock('c:crosses', 'val', 'autoZero');
        }
        if (isset($this->options['radar']) || ($type == 'cat' && $style->showGridX()) || ($type == 'val' && $style->showGridY())) {
            $xmlWriter->writeElement('c:majorGridlines');
        }

        $xmlWriter->startElement('c:scaling');
        $xmlWriter->writeElementBlock('c:orientation', 'val', 'minMax');
        $xmlWriter->endElement(); // c:scaling

        $this->writeShape($xmlWriter, true);

        $xmlWriter->endElement(); // $axisType
    }

    /**
     * Write shape
     *
     * @see  http://www.datypic.com/sc/ooxml/t-a_CT_ShapeProperties.html
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param bool $line
     */
    private function writeShape(XMLWriter $xmlWriter, $line = false)
    {
        $xmlWriter->startElement('c:spPr');
        $xmlWriter->startElement('a:ln');
        if ($line === true) {
            $xmlWriter->writeElement('a:solidFill');
        } else {
            $xmlWriter->writeElement('a:noFill');
        }
        $xmlWriter->endElement(); // a:ln
        $xmlWriter->endElement(); // c:spPr
    }

    private function writeAxisTitle(XMLWriter $xmlWriter, $title)
    {
        $xmlWriter->startElement('c:title'); //start c:title
        $xmlWriter->startElement('c:tx'); //start c:tx
        $xmlWriter->startElement('c:rich'); // start c:rich
        $xmlWriter->writeElement('a:bodyPr');
        $xmlWriter->writeElement('a:lstStyle');
        $xmlWriter->startElement('a:p');
        $xmlWriter->startElement('a:pPr');
        $xmlWriter->writeElement('a:defRPr');
        $xmlWriter->endElement(); // end a:pPr
        $xmlWriter->startElement('a:r');
        $xmlWriter->writeElementBlock('a:rPr', 'lang', 'en-US');

        $xmlWriter->startElement('a:t');
        $xmlWriter->writeRaw($title);
        $xmlWriter->endElement(); //end a:t

        $xmlWriter->endElement(); // end a:r
        $xmlWriter->endElement(); //end a:p
        $xmlWriter->endElement(); //end c:rich
        $xmlWriter->endElement(); // end c:tx
        $xmlWriter->writeElementBlock('c:overlay', 'val', '0');
        $xmlWriter->endElement(); // end c:title
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Comments.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Element\Comment;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Writer\Word2007\Element\Container;

/**
 * Word2007 comments part writer: word/comments.xml
 */
class Comments extends AbstractPart
{
    /**
     * Comments collection to be written
     *
     * @var \PhpOffice\PhpWord\Element\Comment[]
     */
    protected $elements;

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:comments');
        $xmlWriter->writeAttribute('xmlns:ve', 'http://schemas.openxmlformats.org/markup-compatibility/2006');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:wp', 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing');
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:wne', 'http://schemas.microsoft.com/office/word/2006/wordml');

        if ($this->elements !== null) {
            foreach ($this->elements as $element) {
                if ($element instanceof Comment) {
                    $this->writeComment($xmlWriter, $element);
                }
            }
        }

        $xmlWriter->endElement(); // w:comments

        return $xmlWriter->getData();
    }

    /**
     * Write comment item.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Comment $comment
     */
    protected function writeComment(XMLWriter $xmlWriter, Comment $comment)
    {
        $xmlWriter->startElement('w:comment');
        $xmlWriter->writeAttribute('w:id', $comment->getElementId());
        $xmlWriter->writeAttribute('w:author', $comment->getAuthor());
        if ($comment->getDate() != null) {
            $xmlWriter->writeAttribute('w:date', $comment->getDate()->format($this->dateFormat));
        }
        $xmlWriter->writeAttributeIf($comment->getInitials() != null, 'w:initials', $comment->getInitials());

        $containerWriter = new Container($xmlWriter, $comment);
        $containerWriter->write();

        $xmlWriter->endElement(); // w:comment
    }

    /**
     * Set element
     *
     * @param \PhpOffice\PhpWord\Element\Comment[] $elements
     * @return self
     */
    public function setElements($elements)
    {
        $this->elements = $elements;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/ContentTypes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Word2007 contenttypes part writer: [Content_Types].xml
 */
class ContentTypes extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        /** @var \PhpOffice\PhpWord\Writer\Word2007 $parentWriter Type hint */
        $parentWriter = $this->getParentWriter();
        $contentTypes = $parentWriter->getContentTypes();

        $openXMLPrefix = 'application/vnd.openxmlformats-';
        $wordMLPrefix = $openXMLPrefix . 'officedocument.wordprocessingml.';
        $drawingMLPrefix = $openXMLPrefix . 'officedocument.drawingml.';
        $overrides = array(
            '/docProps/core.xml'     => $openXMLPrefix . 'package.core-properties+xml',
            '/docProps/app.xml'      => $openXMLPrefix . 'officedocument.extended-properties+xml',
            '/docProps/custom.xml'   => $openXMLPrefix . 'officedocument.custom-properties+xml',
            '/word/document.xml'     => $wordMLPrefix . 'document.main+xml',
            '/word/styles.xml'       => $wordMLPrefix . 'styles+xml',
            '/word/numbering.xml'    => $wordMLPrefix . 'numbering+xml',
            '/word/settings.xml'     => $wordMLPrefix . 'settings+xml',
            '/word/theme/theme1.xml' => $openXMLPrefix . 'officedocument.theme+xml',
            '/word/webSettings.xml'  => $wordMLPrefix . 'webSettings+xml',
            '/word/fontTable.xml'    => $wordMLPrefix . 'fontTable+xml',
            '/word/comments.xml'     => $wordMLPrefix . 'comments+xml',
        );

        $defaults = $contentTypes['default'];
        if (!empty($contentTypes['override'])) {
            foreach ($contentTypes['override'] as $key => $val) {
                if ($val == 'chart') {
                    $overrides[$key] = $drawingMLPrefix . $val . '+xml';
                } else {
                    $overrides[$key] = $wordMLPrefix . $val . '+xml';
                }
            }
        }

        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('Types');
        $xmlWriter->writeAttribute('xmlns', 'http://schemas.openxmlformats.org/package/2006/content-types');

        $this->writeContentType($xmlWriter, $defaults, true);
        $this->writeContentType($xmlWriter, $overrides, false);

        $xmlWriter->endElement(); // Types

        return $xmlWriter->getData();
    }

    /**
     * Write content types element
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter XML Writer
     * @param array $parts
     * @param bool $isDefault
     */
    private function writeContentType(XMLWriter $xmlWriter, $parts, $isDefault)
    {
        foreach ($parts as $partName => $contentType) {
            $partType = $isDefault ? 'Default' : 'Override';
            $partAttribute = $isDefault ? 'Extension' : 'PartName';
            $xmlWriter->startElement($partType);
            $xmlWriter->writeAttribute($partAttribute, $partName);
            $xmlWriter->writeAttribute('ContentType', $contentType);
            $xmlWriter->endElement();
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/DocPropsApp.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 extended document properties part writer: docProps/app.xml
 *
 * @since 0.11.0
 */
class DocPropsApp extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $xmlWriter = $this->getXmlWriter();
        $schema = 'http://schemas.openxmlformats.org/officeDocument/2006/extended-properties';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('Properties');
        $xmlWriter->writeAttribute('xmlns', $schema);
        $xmlWriter->writeAttribute('xmlns:vt', 'http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes');

        $xmlWriter->writeElement('Application', 'PHPWord');
        $xmlWriter->writeElement('Company', $phpWord->getDocInfo()->getCompany());
        $xmlWriter->writeElement('Manager', $phpWord->getDocInfo()->getManager());

        $xmlWriter->endElement(); // Properties

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/DocPropsCore.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 core document properties part writer: docProps/core.xml
 *
 * @since 0.11.0
 */
class DocPropsCore extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $xmlWriter = $this->getXmlWriter();
        $schema = 'http://schemas.openxmlformats.org/package/2006/metadata/core-properties';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('cp:coreProperties');
        $xmlWriter->writeAttribute('xmlns:cp', $schema);
        $xmlWriter->writeAttribute('xmlns:dc', 'http://purl.org/dc/elements/1.1/');
        $xmlWriter->writeAttribute('xmlns:dcterms', 'http://purl.org/dc/terms/');
        $xmlWriter->writeAttribute('xmlns:dcmitype', 'http://purl.org/dc/dcmitype/');
        $xmlWriter->writeAttribute('xmlns:xsi', 'http://www.w3.org/2001/XMLSchema-instance');

        $xmlWriter->writeElement('dc:creator', $phpWord->getDocInfo()->getCreator());
        $xmlWriter->writeElement('dc:title', $phpWord->getDocInfo()->getTitle());
        $xmlWriter->writeElement('dc:description', $phpWord->getDocInfo()->getDescription());
        $xmlWriter->writeElement('dc:subject', $phpWord->getDocInfo()->getSubject());
        $xmlWriter->writeElement('cp:keywords', $phpWord->getDocInfo()->getKeywords());
        $xmlWriter->writeElement('cp:category', $phpWord->getDocInfo()->getCategory());
        $xmlWriter->writeElement('cp:lastModifiedBy', $phpWord->getDocInfo()->getLastModifiedBy());

        // dcterms:created
        $xmlWriter->startElement('dcterms:created');
        $xmlWriter->writeAttribute('xsi:type', 'dcterms:W3CDTF');
        $xmlWriter->text(date($this->dateFormat, $phpWord->getDocInfo()->getCreated()));
        $xmlWriter->endElement();

        // dcterms:modified
        $xmlWriter->startElement('dcterms:modified');
        $xmlWriter->writeAttribute('xsi:type', 'dcterms:W3CDTF');
        $xmlWriter->text(date($this->dateFormat, $phpWord->getDocInfo()->getModified()));
        $xmlWriter->endElement();

        $xmlWriter->endElement(); // cp:coreProperties

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/DocPropsCustom.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 custom document properties part writer: docProps/custom.xml
 *
 * @since 0.11.0
 */
class DocPropsCustom extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('Properties');
        $xmlWriter->writeAttribute('xmlns', 'http://schemas.openxmlformats.org/officeDocument/2006/custom-properties');
        $xmlWriter->writeAttribute('xmlns:vt', 'http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes');

        $docProps = $phpWord->getDocInfo();
        $properties = $docProps->getCustomProperties();
        foreach ($properties as $key => $property) {
            $propertyValue = $docProps->getCustomPropertyValue($property);
            $propertyType = $docProps->getCustomPropertyType($property);

            $xmlWriter->startElement('property');
            $xmlWriter->writeAttribute('fmtid', '{D5CDD505-2E9C-101B-9397-08002B2CF9AE}');
            $xmlWriter->writeAttribute('pid', $key + 2);
            $xmlWriter->writeAttribute('name', $property);
            switch ($propertyType) {
                case 'i':
                    $xmlWriter->writeElement('vt:i4', $propertyValue);
                    break;
                case 'f':
                    $xmlWriter->writeElement('vt:r8', $propertyValue);
                    break;
                case 'b':
                    $xmlWriter->writeElement('vt:bool', ($propertyValue) ? 'true' : 'false');
                    break;
                case 'd':
                    if ($propertyValue instanceof \DateTime) {
                        $xmlWriter->writeElement('vt:filetime', $propertyValue->format($this->dateFormat));
                    } else {
                        $xmlWriter->writeElement('vt:filetime', date($this->dateFormat, $propertyValue));
                    }
                    break;
                default:
                    $xmlWriter->writeElement('vt:lpwstr', $propertyValue);
                    break;
            }
            $xmlWriter->endElement(); // property
        }

        $xmlWriter->endElement(); // Properties

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Document.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Element\Section;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Writer\Word2007\Element\Container;
use PhpOffice\PhpWord\Writer\Word2007\Style\Section as SectionStyleWriter;

/**
 * Word2007 document part writer: word/document.xml
 */
class Document extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $xmlWriter = $this->getXmlWriter();

        $sections = $phpWord->getSections();
        $sectionCount = count($sections);
        $currentSection = 0;
        $drawingSchema = 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:document');
        $xmlWriter->writeAttribute('xmlns:ve', 'http://schemas.openxmlformats.org/markup-compatibility/2006');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:wp', $drawingSchema);
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:wne', 'http://schemas.microsoft.com/office/word/2006/wordml');

        $xmlWriter->startElement('w:body');

        if ($sectionCount > 0) {
            foreach ($sections as $section) {
                $currentSection++;

                $containerWriter = new Container($xmlWriter, $section);
                $containerWriter->write();

                if ($currentSection == $sectionCount) {
                    $this->writeSectionSettings($xmlWriter, $section);
                } else {
                    $this->writeSection($xmlWriter, $section);
                }
            }
        }

        $xmlWriter->endElement(); // w:body
        $xmlWriter->endElement(); // w:document

        return $xmlWriter->getData();
    }

    /**
     * Write begin section.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Section $section
     */
    private function writeSection(XMLWriter $xmlWriter, Section $section)
    {
        $xmlWriter->startElement('w:p');
        $xmlWriter->startElement('w:pPr');
        $this->writeSectionSettings($xmlWriter, $section);
        $xmlWriter->endElement();
        $xmlWriter->endElement();
    }

    /**
     * Write end section.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Section $section
     */
    private function writeSectionSettings(XMLWriter $xmlWriter, Section $section)
    {
        $xmlWriter->startElement('w:sectPr');

        // Header reference
        foreach ($section->getHeaders() as $header) {
            $rId = $header->getRelationId();
            $xmlWriter->startElement('w:headerReference');
            $xmlWriter->writeAttribute('w:type', $header->getType());
            $xmlWriter->writeAttribute('r:id', 'rId' . $rId);
            $xmlWriter->endElement();
        }

        // Footer reference
        foreach ($section->getFooters() as $footer) {
            $rId = $footer->getRelationId();
            $xmlWriter->startElement('w:footerReference');
            $xmlWriter->writeAttribute('w:type', $footer->getType());
            $xmlWriter->writeAttribute('r:id', 'rId' . $rId);
            $xmlWriter->endElement();
        }

        // Different first page
        if ($section->hasDifferentFirstPage()) {
            $xmlWriter->startElement('w:titlePg');
            $xmlWriter->endElement();
        }

        // Footnote properties
        if ($section->getFootnoteProperties() !== null) {
            $xmlWriter->startElement('w:footnotePr');
            if ($section->getFootnoteProperties()->getPos() != null) {
                $xmlWriter->startElement('w:pos');
                $xmlWriter->writeAttribute('w:val', $section->getFootnoteProperties()->getPos());
                $xmlWriter->endElement();
            }
            if ($section->getFootnoteProperties()->getNumFmt() != null) {
                $xmlWriter->startElement('w:numFmt');
                $xmlWriter->writeAttribute('w:val', $section->getFootnoteProperties()->getNumFmt());
                $xmlWriter->endElement();
            }
            if ($section->getFootnoteProperties()->getNumStart() != null) {
                $xmlWriter->startElement('w:numStart');
                $xmlWriter->writeAttribute('w:val', $section->getFootnoteProperties()->getNumStart());
                $xmlWriter->endElement();
            }
            if ($section->getFootnoteProperties()->getNumRestart() != null) {
                $xmlWriter->startElement('w:numRestart');
                $xmlWriter->writeAttribute('w:val', $section->getFootnoteProperties()->getNumRestart());
                $xmlWriter->endElement();
            }
            $xmlWriter->endElement();
        }

        // Section settings
        $styleWriter = new SectionStyleWriter($xmlWriter, $section->getStyle());
        $styleWriter->write();

        $xmlWriter->endElement(); // w:sectPr
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Endnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 endnotes part writer: word/endnotes.xml
 */
class Endnotes extends Footnotes
{
    /**
     * Name of XML root element
     *
     * @var string
     */
    protected $rootNode = 'w:endnotes';

    /**
     * Name of XML node element
     *
     * @var string
     */
    protected $elementNode = 'w:endnote';

    /**
     * Name of XML reference element
     *
     * @var string
     */
    protected $refNode = 'w:endnoteRef';

    /**
     * Reference style name
     *
     * @var string
     */
    protected $refStyle = 'EndnoteReference';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/FontTable.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 font table writer: word/fontTable.xml
 *
 * @todo Generate content dynamically
 * @since 0.10.0
 */
class FontTable extends AbstractPart
{
    /**
     * Write fontTable.xml.
     *
     * @return string
     */
    public function write()
    {
        $str = '';
        $str .= '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
        $str .= '<w:fonts ' .
            'xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" ' .
            'xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">';

        $str .= '<w:font w:name="Times New Roman">';
        $str .= '<w:panose1 w:val="02020603050405020304" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="roman" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="E0002AFF" w:usb1="C0007841" w:usb2="00000009" w:usb3="00000000" ' .
            'w:csb0="000001FF" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Courier New">';
        $str .= '<w:panose1 w:val="02070309020205020404" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="modern" />';
        $str .= '<w:pitch w:val="fixed" />';
        $str .= '<w:sig w:usb0="E0002AFF" w:usb1="C0007843" w:usb2="00000009" w:usb3="00000000" ' .
            'w:csb0="000001FF" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Wingdings">';
        $str .= '<w:panose1 w:val="05000000000000000000" />';
        $str .= '<w:charset w:val="02" />';
        $str .= '<w:family w:val="auto" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="00000000" w:usb1="10000000" w:usb2="00000000" w:usb3="00000000" ' .
            'w:csb0="80000000" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Symbol">';
        $str .= '<w:panose1 w:val="05050102010706020507" />';
        $str .= '<w:charset w:val="02" />';
        $str .= '<w:family w:val="roman" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="00000000" w:usb1="10000000" w:usb2="00000000" w:usb3="00000000" ' .
            'w:csb0="80000000" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Arial">';
        $str .= '<w:panose1 w:val="020B0604020202020204" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="swiss" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="E0002AFF" w:usb1="C0007843" w:usb2="00000009" w:usb3="00000000" ' .
            'w:csb0="000001FF" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Cambria">';
        $str .= '<w:panose1 w:val="02040503050406030204" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="roman" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="A00002EF" w:usb1="4000004B" w:usb2="00000000" w:usb3="00000000" ' .
            'w:csb0="0000019F" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Calibri">';
        $str .= '<w:panose1 w:val="020F0502020204030204" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="swiss" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="E10002FF" w:usb1="4000ACFF" w:usb2="00000009" w:usb3="00000000" ' .
            'w:csb0="0000019F" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '<w:font w:name="Garamond">';
        $str .= '<w:panose1 w:val="02020404030301010803" />';
        $str .= '<w:charset w:val="00" />';
        $str .= '<w:family w:val="roman" />';
        $str .= '<w:pitch w:val="variable" />';
        $str .= '<w:sig w:usb0="00000287" w:usb1="00000002" w:usb2="00000000" w:usb3="00000000" ' .
            'w:csb0="0000009F" w:csb1="00000000" />';
        $str .= '</w:font>';

        $str .= '</w:fonts>';

        return $str;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Footer.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Writer\Word2007\Element\Container;

/**
 * Word2007 footer part writer: word/footerx.xml
 */
class Footer extends AbstractPart
{
    /**
     * Root element name
     *
     * @var string
     */
    protected $rootElement = 'w:ftr';

    /**
     * Footer/header element to be written
     *
     * @var \PhpOffice\PhpWord\Element\Footer
     */
    protected $element;

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $drawingSchema = 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement($this->rootElement);
        $xmlWriter->writeAttribute('xmlns:ve', 'http://schemas.openxmlformats.org/markup-compatibility/2006');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:wp', $drawingSchema);
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:wne', 'http://schemas.microsoft.com/office/word/2006/wordml');

        $containerWriter = new Container($xmlWriter, $this->element);
        $containerWriter->write();

        $xmlWriter->endElement(); // $this->rootElement

        return $xmlWriter->getData();
    }

    /**
     * Set element
     *
     * @param \PhpOffice\PhpWord\Element\Footer|\PhpOffice\PhpWord\Element\Header $element
     * @return self
     */
    public function setElement($element)
    {
        $this->element = $element;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Footnotes.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Element\Footnote;
use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Writer\Word2007\Element\Container;
use PhpOffice\PhpWord\Writer\Word2007\Style\Paragraph as ParagraphStyleWriter;

/**
 * Word2007 footnotes part writer: word/(footnotes|endnotes).xml
 */
class Footnotes extends AbstractPart
{
    /**
     * Name of XML root element
     *
     * @var string
     */
    protected $rootNode = 'w:footnotes';

    /**
     * Name of XML node element
     *
     * @var string
     */
    protected $elementNode = 'w:footnote';

    /**
     * Name of XML reference element
     *
     * @var string
     */
    protected $refNode = 'w:footnoteRef';

    /**
     * Reference style name
     *
     * @var string
     */
    protected $refStyle = 'FootnoteReference';

    /**
     * Footnotes/endnotes collection to be written
     *
     * @var \PhpOffice\PhpWord\Collection\Footnotes|\PhpOffice\PhpWord\Collection\Endnotes
     */
    protected $elements;

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $drawingSchema = 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement($this->rootNode);
        $xmlWriter->writeAttribute('xmlns:ve', 'http://schemas.openxmlformats.org/markup-compatibility/2006');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:wp', $drawingSchema);
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:wne', 'http://schemas.microsoft.com/office/word/2006/wordml');

        // Separator and continuation separator
        $xmlWriter->startElement($this->elementNode);
        $xmlWriter->writeAttribute('w:id', -1);
        $xmlWriter->writeAttribute('w:type', 'separator');
        $xmlWriter->startElement('w:p');
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:separator');
        $xmlWriter->endElement(); // w:separator
        $xmlWriter->endElement(); // w:r
        $xmlWriter->endElement(); // w:p
        $xmlWriter->endElement(); // $this->elementNode
        $xmlWriter->startElement($this->elementNode);
        $xmlWriter->writeAttribute('w:id', 0);
        $xmlWriter->writeAttribute('w:type', 'continuationSeparator');
        $xmlWriter->startElement('w:p');
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:continuationSeparator');
        $xmlWriter->endElement(); // w:continuationSeparator
        $xmlWriter->endElement(); // w:r
        $xmlWriter->endElement(); // w:p
        $xmlWriter->endElement(); // $this->elementNode

        /** @var array $elements Type hint */
        $elements = $this->elements;
        foreach ($elements as $element) {
            if ($element instanceof Footnote) {
                $this->writeNote($xmlWriter, $element);
            }
        }

        $xmlWriter->endElement(); // $this->rootNode

        return $xmlWriter->getData();
    }

    /**
     * Set element
     *
     * @param \PhpOffice\PhpWord\Collection\Footnotes|\PhpOffice\PhpWord\Collection\Endnotes $elements
     * @return self
     */
    public function setElements($elements)
    {
        $this->elements = $elements;

        return $this;
    }

    /**
     * Write note item.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Element\Footnote|\PhpOffice\PhpWord\Element\Endnote $element
     */
    protected function writeNote(XMLWriter $xmlWriter, $element)
    {
        $xmlWriter->startElement($this->elementNode);
        $xmlWriter->writeAttribute('w:id', $element->getRelationId());
        $xmlWriter->startElement('w:p');

        // Paragraph style
        $styleWriter = new ParagraphStyleWriter($xmlWriter, $element->getParagraphStyle());
        $styleWriter->setIsInline(true);
        $styleWriter->write();

        // Reference symbol
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:rPr');
        $xmlWriter->startElement('w:rStyle');
        $xmlWriter->writeAttribute('w:val', $this->refStyle);
        $xmlWriter->endElement(); // w:rStyle
        $xmlWriter->endElement(); // w:rPr
        $xmlWriter->writeElement($this->refNode);
        $xmlWriter->endElement(); // w:r

        // Empty space after refence symbol
        $xmlWriter->startElement('w:r');
        $xmlWriter->startElement('w:t');
        $xmlWriter->writeAttribute('xml:space', 'preserve');
        $xmlWriter->text(' ');
        $xmlWriter->endElement(); // w:t
        $xmlWriter->endElement(); // w:r

        $containerWriter = new Container($xmlWriter, $element);
        $containerWriter->write();

        $xmlWriter->endElement(); // w:p
        $xmlWriter->endElement(); // $this->elementNode
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Header.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 header part writer: word/headerx.xml
 */
class Header extends Footer
{
    /**
     * Root element name
     *
     * @var string
     */
    protected $rootElement = 'w:hdr';
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Numbering.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Numbering as NumberingStyle;
use PhpOffice\PhpWord\Style\NumberingLevel;

/**
 * Word2007 numbering part writer: word/numbering.xml
 *
 * @since 0.10.0
 */
class Numbering extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $styles = Style::getStyles();
        $drawingSchema = 'http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing';

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:numbering');
        $xmlWriter->writeAttribute('xmlns:ve', 'http://schemas.openxmlformats.org/markup-compatibility/2006');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:wp', $drawingSchema);
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:wne', 'http://schemas.microsoft.com/office/word/2006/wordml');

        // Abstract numbering definitions
        foreach ($styles as $style) {
            if ($style instanceof NumberingStyle) {
                $levels = $style->getLevels();

                $xmlWriter->startElement('w:abstractNum');
                $xmlWriter->writeAttribute('w:abstractNumId', $style->getIndex());

                $xmlWriter->startElement('w:nsid');
                $xmlWriter->writeAttribute('w:val', $this->getRandomHexNumber());
                $xmlWriter->endElement(); // w:nsid

                $xmlWriter->startElement('w:multiLevelType');
                $xmlWriter->writeAttribute('w:val', $style->getType());
                $xmlWriter->endElement(); // w:multiLevelType

                if (is_array($levels)) {
                    foreach ($levels as $level) {
                        $this->writeLevel($xmlWriter, $level);
                    }
                }
                $xmlWriter->endElement(); // w:abstractNum
            }
        }

        // Numbering definition instances
        foreach ($styles as $style) {
            if ($style instanceof NumberingStyle) {
                $xmlWriter->startElement('w:num');
                $xmlWriter->writeAttribute('w:numId', $style->getIndex());
                $xmlWriter->startElement('w:abstractNumId');
                $xmlWriter->writeAttribute('w:val', $style->getIndex());
                $xmlWriter->endElement(); // w:abstractNumId
                $xmlWriter->endElement(); // w:num
            }
        }

        $xmlWriter->endElement(); // w:numbering

        return $xmlWriter->getData();
    }

    /**
     * Write level.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\NumberingLevel $level
     */
    private function writeLevel(XMLWriter $xmlWriter, NumberingLevel $level)
    {
        $xmlWriter->startElement('w:lvl');
        $xmlWriter->writeAttribute('w:ilvl', $level->getLevel());

        // Numbering level properties
        $properties = array(
            'start'     => 'start',
            'format'    => 'numFmt',
            'restart'   => 'lvlRestart',
            'pStyle'    => 'pStyle',
            'suffix'    => 'suff',
            'text'      => 'lvlText',
            'alignment' => 'lvlJc',
        );
        foreach ($properties as $property => $nodeName) {
            $getMethod = "get{$property}";
            if ('' !== $level->$getMethod()         // this condition is now supported by `alignment` only
                && !is_null($level->$getMethod())) {
                $xmlWriter->startElement("w:{$nodeName}");
                $xmlWriter->writeAttribute('w:val', $level->$getMethod());
                $xmlWriter->endElement(); // w:start
            }
        }

        // Paragraph & font styles
        $this->writeParagraph($xmlWriter, $level);
        $this->writeFont($xmlWriter, $level);

        $xmlWriter->endElement(); // w:lvl
    }

    /**
     * Write level paragraph.
     *
     * @since 0.11.0
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\NumberingLevel $level
     * @todo Use paragraph style writer
     */
    private function writeParagraph(XMLWriter $xmlWriter, NumberingLevel $level)
    {
        $tabPos = $level->getTabPos();
        $left = $level->getLeft();
        $hanging = $level->getHanging();

        $xmlWriter->startElement('w:pPr');

        $xmlWriter->startElement('w:tabs');
        $xmlWriter->startElement('w:tab');
        $xmlWriter->writeAttribute('w:val', 'num');
        $xmlWriter->writeAttributeIf($tabPos !== null, 'w:pos', $tabPos);
        $xmlWriter->endElement(); // w:tab
        $xmlWriter->endElement(); // w:tabs

        $xmlWriter->startElement('w:ind');
        $xmlWriter->writeAttributeIf($left !== null, 'w:left', $left);
        $xmlWriter->writeAttributeIf($hanging !== null, 'w:hanging', $hanging);
        $xmlWriter->endElement(); // w:ind

        $xmlWriter->endElement(); // w:pPr
    }

    /**
     * Write level font.
     *
     * @since 0.11.0
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\NumberingLevel $level
     * @todo Use font style writer
     */
    private function writeFont(XMLWriter $xmlWriter, NumberingLevel $level)
    {
        $font = $level->getFont();
        $hint = $level->getHint();

        $xmlWriter->startElement('w:rPr');
        $xmlWriter->startElement('w:rFonts');
        $xmlWriter->writeAttributeIf($font !== null, 'w:ascii', $font);
        $xmlWriter->writeAttributeIf($font !== null, 'w:hAnsi', $font);
        $xmlWriter->writeAttributeIf($font !== null, 'w:cs', $font);
        $xmlWriter->writeAttributeIf($hint !== null, 'w:hint', $hint);
        $xmlWriter->endElement(); // w:rFonts
        $xmlWriter->endElement(); // w:rPr
    }

    /**
     * Get random hexadecimal number value
     *
     * @param int $length
     * @return string
     */
    private function getRandomHexNumber($length = 8)
    {
        return strtoupper(substr(md5(rand()), 0, $length));
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Rels.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Exception\Exception;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Word2007 main relationship writer: _rels/.rels
 *
 * @since 0.10.0
 */
class Rels extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlRels = array(
            'docProps/core.xml'   => 'package/2006/relationships/metadata/core-properties',
            'docProps/app.xml'    => 'officeDocument/2006/relationships/extended-properties',
            'docProps/custom.xml' => 'officeDocument/2006/relationships/custom-properties',
            'word/document.xml'   => 'officeDocument/2006/relationships/officeDocument',
        );
        $xmlWriter = $this->getXmlWriter();
        $this->writeRels($xmlWriter, $xmlRels);

        return $xmlWriter->getData();
    }

    /**
     * Write relationships.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param array $xmlRels
     * @param array $mediaRels
     * @param int $relId
     */
    protected function writeRels(XMLWriter $xmlWriter, $xmlRels = array(), $mediaRels = array(), $relId = 1)
    {
        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('Relationships');
        $xmlWriter->writeAttribute('xmlns', 'http://schemas.openxmlformats.org/package/2006/relationships');

        // XML files relationships
        foreach ($xmlRels as $target => $type) {
            $this->writeRel($xmlWriter, $relId++, $type, $target);
        }

        // Media relationships
        foreach ($mediaRels as $mediaRel) {
            $this->writeMediaRel($xmlWriter, $relId++, $mediaRel);
        }

        $xmlWriter->endElement(); // Relationships
    }

    /**
     * Write media relationships.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param int $relId
     * @param array $mediaRel
     */
    private function writeMediaRel(XMLWriter $xmlWriter, $relId, $mediaRel)
    {
        $typePrefix = 'officeDocument/2006/relationships/';
        $typeMapping = array('image' => 'image', 'object' => 'oleObject', 'link' => 'hyperlink');
        $targetMapping = array('image' => 'media/', 'object' => 'embeddings/');

        $mediaType = $mediaRel['type'];
        $type = isset($typeMapping[$mediaType]) ? $typeMapping[$mediaType] : $mediaType;
        $targetPrefix = isset($targetMapping[$mediaType]) ? $targetMapping[$mediaType] : '';
        $target = $mediaRel['target'];
        $targetMode = ($type == 'hyperlink') ? 'External' : '';

        $this->writeRel($xmlWriter, $relId, $typePrefix . $type, $targetPrefix . $target, $targetMode);
    }

    /**
     * Write individual rels entry.
     *
     * Format:
     * <Relationship Id="rId..." Type="http://..." Target="....xml" TargetMode="..." />
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param int $relId Relationship ID
     * @param string $type Relationship type
     * @param string $target Relationship target
     * @param string $targetMode Relationship target mode
     *
     * @throws \PhpOffice\PhpWord\Exception\Exception
     */
    private function writeRel(XMLWriter $xmlWriter, $relId, $type, $target, $targetMode = '')
    {
        if ($type != '' && $target != '') {
            if (strpos($relId, 'rId') === false) {
                $relId = 'rId' . $relId;
            }
            $xmlWriter->startElement('Relationship');
            $xmlWriter->writeAttribute('Id', $relId);
            $xmlWriter->writeAttribute('Type', 'http://schemas.openxmlformats.org/' . $type);
            $xmlWriter->writeAttribute('Target', $target);
            if ($targetMode != '') {
                $xmlWriter->writeAttribute('TargetMode', $targetMode);
            }
            $xmlWriter->endElement();
        } else {
            throw new Exception('Invalid parameters passed.');
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/RelsDocument.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 document relationship writer: word/_rels/document.xml.rels
 *
 * @since 0.11.0
 */
class RelsDocument extends Rels
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlRels = array(
            'styles.xml'       => 'officeDocument/2006/relationships/styles',
            'numbering.xml'    => 'officeDocument/2006/relationships/numbering',
            'settings.xml'     => 'officeDocument/2006/relationships/settings',
            'theme/theme1.xml' => 'officeDocument/2006/relationships/theme',
            'webSettings.xml'  => 'officeDocument/2006/relationships/webSettings',
            'fontTable.xml'    => 'officeDocument/2006/relationships/fontTable',
        );
        $xmlWriter = $this->getXmlWriter();

        /** @var \PhpOffice\PhpWord\Writer\Word2007 $parentWriter Type hint */
        $parentWriter = $this->getParentWriter();
        $this->writeRels($xmlWriter, $xmlRels, $parentWriter->getRelationships());

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/RelsPart.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 part relationship writer: word/_rels/(header|footer|footnotes|endnotes)*.xml.rels
 *
 * @since 0.11.0
 */
class RelsPart extends Rels
{
    /**
     * Media relationships
     *
     * @var array
     */
    private $media = array();

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();
        $this->writeRels($xmlWriter, array(), $this->media);

        return $xmlWriter->getData();
    }

    /**
     * Set media
     *
     * @param array $media
     * @return self
     */
    public function setMedia($media)
    {
        $this->media = $media;

        return $this;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Settings.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\ComplexType\ProofState;
use PhpOffice\PhpWord\ComplexType\TrackChangesView;
use PhpOffice\PhpWord\Shared\Microsoft\PasswordEncoder;
use PhpOffice\PhpWord\Style\Language;

/**
 * Word2007 settings part writer: word/settings.xml
 *
 * @see  http://www.schemacentral.com/sc/ooxml/t-w_CT_Settings.html
 */
class Settings extends AbstractPart
{
    /**
     * Settings value
     *
     * @var array
     */
    private $settings = array();

    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $this->getSettings();

        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:settings');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');
        $xmlWriter->writeAttribute('xmlns:m', 'http://schemas.openxmlformats.org/officeDocument/2006/math');
        $xmlWriter->writeAttribute('xmlns:sl', 'http://schemas.openxmlformats.org/schemaLibrary/2006/main');
        $xmlWriter->writeAttribute('xmlns:o', 'urn:schemas-microsoft-com:office:office');
        $xmlWriter->writeAttribute('xmlns:v', 'urn:schemas-microsoft-com:vml');
        $xmlWriter->writeAttribute('xmlns:w10', 'urn:schemas-microsoft-com:office:word');

        foreach ($this->settings as $settingKey => $settingValue) {
            $this->writeSetting($xmlWriter, $settingKey, $settingValue);
        }

        $xmlWriter->endElement(); // w:settings

        return $xmlWriter->getData();
    }

    /**
     * Write indivual setting, recursive to any child settings.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $settingKey
     * @param array|string $settingValue
     */
    protected function writeSetting($xmlWriter, $settingKey, $settingValue)
    {
        if ($settingValue == '') {
            $xmlWriter->writeElement($settingKey);
        } elseif (is_array($settingValue) && !empty($settingValue)) {
            $xmlWriter->startElement($settingKey);

            /** @var array $settingValue Type hint */
            foreach ($settingValue as $childKey => $childValue) {
                if ($childKey == '@attributes') {
                    foreach ($childValue as $key => $val) {
                        $xmlWriter->writeAttribute($key, $val);
                    }
                } else {
                    $this->writeSetting($xmlWriter, $childKey, $childValue);
                }
            }
            $xmlWriter->endElement();
        }
    }

    /**
     * Get settings.
     */
    private function getSettings()
    {
        /** @var \PhpOffice\PhpWord\Metadata\Settings $documentSettings */
        $documentSettings = $this->getParentWriter()->getPhpWord()->getSettings();

        // Default settings
        $this->settings = array(
            'w:defaultTabStop'          => array('@attributes' => array('w:val' => '708')),
            'w:hyphenationZone'         => array('@attributes' => array('w:val' => '425')),
            'w:characterSpacingControl' => array('@attributes' => array('w:val' => 'doNotCompress')),
            'w:decimalSymbol'           => array('@attributes' => array('w:val' => $documentSettings->getDecimalSymbol())),
            'w:listSeparator'           => array('@attributes' => array('w:val' => ';')),
            'w:compat'                  => array(),
            'm:mathPr'                  => array(
                'm:mathFont'   => array('@attributes' => array('m:val' => 'Cambria Math')),
                'm:brkBin'     => array('@attributes' => array('m:val' => 'before')),
                'm:brkBinSub'  => array('@attributes' => array('m:val' => '--')),
                'm:smallFrac'  => array('@attributes' => array('m:val' => 'off')),
                'm:dispDef'    => '',
                'm:lMargin'    => array('@attributes' => array('m:val' => '0')),
                'm:rMargin'    => array('@attributes' => array('m:val' => '0')),
                'm:defJc'      => array('@attributes' => array('m:val' => 'centerGroup')),
                'm:wrapIndent' => array('@attributes' => array('m:val' => '1440')),
                'm:intLim'     => array('@attributes' => array('m:val' => 'subSup')),
                'm:naryLim'    => array('@attributes' => array('m:val' => 'undOvr')),
            ),
            'w:clrSchemeMapping' => array(
                '@attributes' => array(
                    'w:bg1'               => 'light1',
                    'w:t1'                => 'dark1',
                    'w:bg2'               => 'light2',
                    'w:t2'                => 'dark2',
                    'w:accent1'           => 'accent1',
                    'w:accent2'           => 'accent2',
                    'w:accent3'           => 'accent3',
                    'w:accent4'           => 'accent4',
                    'w:accent5'           => 'accent5',
                    'w:accent6'           => 'accent6',
                    'w:hyperlink'         => 'hyperlink',
                    'w:followedHyperlink' => 'followedHyperlink',
                ),
            ),
        );

        $this->setOnOffValue('w:mirrorMargins', $documentSettings->hasMirrorMargins());
        $this->setOnOffValue('w:hideSpellingErrors', $documentSettings->hasHideSpellingErrors());
        $this->setOnOffValue('w:hideGrammaticalErrors', $documentSettings->hasHideGrammaticalErrors());
        $this->setOnOffValue('w:trackRevisions', $documentSettings->hasTrackRevisions());
        $this->setOnOffValue('w:doNotTrackMoves', $documentSettings->hasDoNotTrackMoves());
        $this->setOnOffValue('w:doNotTrackFormatting', $documentSettings->hasDoNotTrackFormatting());
        $this->setOnOffValue('w:evenAndOddHeaders', $documentSettings->hasEvenAndOddHeaders());
        $this->setOnOffValue('w:updateFields', $documentSettings->hasUpdateFields());
        $this->setOnOffValue('w:autoHyphenation', $documentSettings->hasAutoHyphenation());
        $this->setOnOffValue('w:doNotHyphenateCaps', $documentSettings->hasDoNotHyphenateCaps());

        $this->setThemeFontLang($documentSettings->getThemeFontLang());
        $this->setRevisionView($documentSettings->getRevisionView());
        $this->setDocumentProtection($documentSettings->getDocumentProtection());
        $this->setProofState($documentSettings->getProofState());
        $this->setZoom($documentSettings->getZoom());
        $this->setConsecutiveHyphenLimit($documentSettings->getConsecutiveHyphenLimit());
        $this->setHyphenationZone($documentSettings->getHyphenationZone());
        $this->setCompatibility();
    }

    /**
     * Adds a boolean attribute to the settings array
     *
     * @param string $settingName
     * @param bool|null $booleanValue
     */
    private function setOnOffValue($settingName, $booleanValue)
    {
        if (!is_bool($booleanValue)) {
            return;
        }

        $value = $booleanValue ? 'true' : 'false';
        $this->settings[$settingName] = array('@attributes' => array('w:val' => $value));
    }

    /**
     * Get protection settings.
     *
     * @param \PhpOffice\PhpWord\Metadata\Protection $documentProtection
     */
    private function setDocumentProtection($documentProtection)
    {
        if ($documentProtection->getEditing() !== null) {
            if ($documentProtection->getPassword() == null) {
                $this->settings['w:documentProtection'] = array(
                    '@attributes' => array(
                        'w:enforcement' => 1,
                        'w:edit'        => $documentProtection->getEditing(),
                    ),
                );
            } else {
                if ($documentProtection->getSalt() == null) {
                    $documentProtection->setSalt(openssl_random_pseudo_bytes(16));
                }
                $passwordHash = PasswordEncoder::hashPassword($documentProtection->getPassword(), $documentProtection->getAlgorithm(), $documentProtection->getSalt(), $documentProtection->getSpinCount());
                $this->settings['w:documentProtection'] = array(
                    '@attributes' => array(
                        'w:enforcement'         => 1,
                        'w:edit'                => $documentProtection->getEditing(),
                        'w:cryptProviderType'   => 'rsaFull',
                        'w:cryptAlgorithmClass' => 'hash',
                        'w:cryptAlgorithmType'  => 'typeAny',
                        'w:cryptAlgorithmSid'   => PasswordEncoder::getAlgorithmId($documentProtection->getAlgorithm()),
                        'w:cryptSpinCount'      => $documentProtection->getSpinCount(),
                        'w:hash'                => $passwordHash,
                        'w:salt'                => base64_encode($documentProtection->getSalt()),
                    ),
                );
            }
        }
    }

    /**
     * Set the Proof state
     *
     * @param ProofState $proofState
     */
    private function setProofState(ProofState $proofState = null)
    {
        if ($proofState != null && $proofState->getGrammar() !== null && $proofState->getSpelling() !== null) {
            $this->settings['w:proofState'] = array(
                '@attributes' => array(
                    'w:spelling' => $proofState->getSpelling(),
                    'w:grammar'  => $proofState->getGrammar(),
                ),
            );
        }
    }

    /**
     * Set the Revision View
     *
     * @param TrackChangesView $trackChangesView
     */
    private function setRevisionView(TrackChangesView $trackChangesView = null)
    {
        if ($trackChangesView != null) {
            $revisionView = array();
            $revisionView['w:markup'] = $trackChangesView->hasMarkup() ? 'true' : 'false';
            $revisionView['w:comments'] = $trackChangesView->hasComments() ? 'true' : 'false';
            $revisionView['w:insDel'] = $trackChangesView->hasInsDel() ? 'true' : 'false';
            $revisionView['w:formatting'] = $trackChangesView->hasFormatting() ? 'true' : 'false';
            $revisionView['w:inkAnnotations'] = $trackChangesView->hasInkAnnotations() ? 'true' : 'false';

            $this->settings['w:revisionView'] = array('@attributes' => $revisionView);
        }
    }

    /**
     * Sets the language
     *
     * @param Language $language
     */
    private function setThemeFontLang(Language $language = null)
    {
        $latinLanguage = ($language == null || $language->getLatin() === null) ? 'en-US' : $language->getLatin();
        $lang = array();
        $lang['w:val'] = $latinLanguage;
        if ($language != null) {
            $lang['w:eastAsia'] = $language->getEastAsia() === null ? 'x-none' : $language->getEastAsia();
            $lang['w:bidi'] = $language->getBidirectional() === null ? 'x-none' : $language->getBidirectional();
        }
        $this->settings['w:themeFontLang'] = array('@attributes' => $lang);
    }

    /**
     * Set the magnification
     *
     * @param mixed $zoom
     */
    private function setZoom($zoom = null)
    {
        if ($zoom !== null) {
            $attr = is_int($zoom) ? 'w:percent' : 'w:val';
            $this->settings['w:zoom'] = array('@attributes' => array($attr => $zoom));
        }
    }

    /**
     * @param int|null $consecutiveHyphenLimit
     */
    private function setConsecutiveHyphenLimit($consecutiveHyphenLimit)
    {
        if ($consecutiveHyphenLimit === null) {
            return;
        }

        $this->settings['w:consecutiveHyphenLimit'] = array(
            '@attributes' => array('w:val' => $consecutiveHyphenLimit),
        );
    }

    /**
     * @param float|null $hyphenationZone
     */
    private function setHyphenationZone($hyphenationZone)
    {
        if ($hyphenationZone === null) {
            return;
        }

        $this->settings['w:hyphenationZone'] = array(
            '@attributes' => array('w:val' => $hyphenationZone),
        );
    }

    /**
     * Get compatibility setting.
     */
    private function setCompatibility()
    {
        $compatibility = $this->getParentWriter()->getPhpWord()->getCompatibility();
        if ($compatibility->getOoxmlVersion() !== null) {
            $this->settings['w:compat']['w:compatSetting'] = array(
                '@attributes' => array(
                    'w:name' => 'compatibilityMode',
                    'w:uri'  => 'http://schemas.microsoft.com/office/word',
                    'w:val'  => $compatibility->getOoxmlVersion(),
                ),
            );
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Styles.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Font as FontStyle;
use PhpOffice\PhpWord\Style\Paragraph as ParagraphStyle;
use PhpOffice\PhpWord\Style\Table as TableStyle;
use PhpOffice\PhpWord\Writer\Word2007\Style\Font as FontStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Paragraph as ParagraphStyleWriter;
use PhpOffice\PhpWord\Writer\Word2007\Style\Table as TableStyleWriter;

/**
 * Word2007 styles part writer: word/styles.xml
 *
 * @todo Do something with the numbering style introduced in 0.10.0
 * @SuppressWarnings(PHPMD.UnusedPrivateMethod) For writeFontStyle, writeParagraphStyle, and writeTableStyle
 */
class Styles extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:styles');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');

        // Write default styles
        $styles = Style::getStyles();
        $this->writeDefaultStyles($xmlWriter, $styles);

        // Write styles
        if (count($styles) > 0) {
            foreach ($styles as $styleName => $style) {
                if ($styleName == 'Normal') {
                    continue;
                }

                // Get style class and execute if the private method exists
                $styleClass = substr(get_class($style), strrpos(get_class($style), '\\') + 1);
                $method = "write{$styleClass}Style";
                if (method_exists($this, $method)) {
                    $this->$method($xmlWriter, $styleName, $style);
                }
            }
        }

        $xmlWriter->endElement(); // w:styles

        return $xmlWriter->getData();
    }

    /**
     * Write default font and other default styles.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\AbstractStyle[] $styles
     */
    private function writeDefaultStyles(XMLWriter $xmlWriter, $styles)
    {
        $phpWord = $this->getParentWriter()->getPhpWord();
        $fontName = $phpWord->getDefaultFontName();
        $fontSize = $phpWord->getDefaultFontSize();
        $language = $phpWord->getSettings()->getThemeFontLang();
        $latinLanguage = ($language == null || $language->getLatin() === null) ? 'en-US' : $language->getLatin();

        // Default font
        $xmlWriter->startElement('w:docDefaults');
        $xmlWriter->startElement('w:rPrDefault');
        $xmlWriter->startElement('w:rPr');
        $xmlWriter->startElement('w:rFonts');
        $xmlWriter->writeAttribute('w:ascii', $fontName);
        $xmlWriter->writeAttribute('w:hAnsi', $fontName);
        $xmlWriter->writeAttribute('w:eastAsia', $fontName);
        $xmlWriter->writeAttribute('w:cs', $fontName);
        $xmlWriter->endElement(); // w:rFonts
        $xmlWriter->startElement('w:sz');
        $xmlWriter->writeAttribute('w:val', $fontSize * 2);
        $xmlWriter->endElement(); // w:sz
        $xmlWriter->startElement('w:szCs');
        $xmlWriter->writeAttribute('w:val', $fontSize * 2);
        $xmlWriter->endElement(); // w:szCs
        $xmlWriter->startElement('w:lang');
        $xmlWriter->writeAttribute('w:val', $latinLanguage);
        if ($language != null) {
            $xmlWriter->writeAttributeIf($language->getEastAsia() !== null, 'w:eastAsia', $language->getEastAsia());
            $xmlWriter->writeAttributeIf($language->getBidirectional() !== null, 'w:bidi', $language->getBidirectional());
        }
        $xmlWriter->endElement(); // w:lang
        $xmlWriter->endElement(); // w:rPr
        $xmlWriter->endElement(); // w:rPrDefault
        $xmlWriter->endElement(); // w:docDefaults

        // Normal style
        $xmlWriter->startElement('w:style');
        $xmlWriter->writeAttribute('w:type', 'paragraph');
        $xmlWriter->writeAttribute('w:default', '1');
        $xmlWriter->writeAttribute('w:styleId', 'Normal');
        $xmlWriter->startElement('w:name');
        $xmlWriter->writeAttribute('w:val', 'Normal');
        $xmlWriter->endElement(); // w:name
        if (isset($styles['Normal'])) {
            $normalStyle = $styles['Normal'];
            // w:pPr
            if ($normalStyle instanceof Fontstyle && $normalStyle->getParagraph() != null) {
                $styleWriter = new ParagraphStyleWriter($xmlWriter, $normalStyle->getParagraph());
                $styleWriter->write();
            } elseif ($normalStyle instanceof ParagraphStyle) {
                $styleWriter = new ParagraphStyleWriter($xmlWriter, $normalStyle);
                $styleWriter->write();
            }

            // w:rPr
            $styleWriter = new FontStyleWriter($xmlWriter, $normalStyle);
            $styleWriter->write();
        }
        $xmlWriter->endElement(); // w:style

        // FootnoteReference style
        if (!isset($styles['FootnoteReference'])) {
            $xmlWriter->startElement('w:style');
            $xmlWriter->writeAttribute('w:type', 'character');
            $xmlWriter->writeAttribute('w:styleId', 'FootnoteReference');
            $xmlWriter->startElement('w:name');
            $xmlWriter->writeAttribute('w:val', 'Footnote Reference');
            $xmlWriter->endElement(); // w:name
            $xmlWriter->writeElement('w:semiHidden');
            $xmlWriter->writeElement('w:unhideWhenUsed');
            $xmlWriter->startElement('w:rPr');
            $xmlWriter->startElement('w:vertAlign');
            $xmlWriter->writeAttribute('w:val', 'superscript');
            $xmlWriter->endElement(); // w:vertAlign
            $xmlWriter->endElement(); // w:rPr
            $xmlWriter->endElement(); // w:style
        }
    }

    /**
     * Write font style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $styleName
     * @param \PhpOffice\PhpWord\Style\Font $style
     */
    private function writeFontStyle(XMLWriter $xmlWriter, $styleName, FontStyle $style)
    {
        $paragraphStyle = $style->getParagraph();
        $styleType = $style->getStyleType();
        $type = ($styleType == 'title') ? 'paragraph' : 'character';
        if (!is_null($paragraphStyle)) {
            $type = 'paragraph';
        }

        $xmlWriter->startElement('w:style');
        $xmlWriter->writeAttribute('w:type', $type);

        // Heading style
        if ($styleType == 'title') {
            $arrStyle = explode('_', $styleName);
            if (count($arrStyle) > 1) {
                $styleId = 'Heading' . $arrStyle[1];
                $styleName = 'heading ' . $arrStyle[1];
                $styleLink = 'Heading' . $arrStyle[1] . 'Char';
            } else {
                $styleId = $styleName;
                $styleName = strtolower($styleName);
                $styleLink = $styleName . 'Char';
            }
            $xmlWriter->writeAttribute('w:styleId', $styleId);

            $xmlWriter->startElement('w:link');
            $xmlWriter->writeAttribute('w:val', $styleLink);
            $xmlWriter->endElement();
        } elseif (!is_null($paragraphStyle)) {
            // if type is 'paragraph' it should have a styleId
            $xmlWriter->writeAttribute('w:styleId', $styleName);
        }

        // Style name
        $xmlWriter->startElement('w:name');
        $xmlWriter->writeAttribute('w:val', $styleName);
        $xmlWriter->endElement();

        // Parent style
        if (!is_null($paragraphStyle)) {
            if ($paragraphStyle->getStyleName() != null) {
                $xmlWriter->writeElementBlock('w:basedOn', 'w:val', $paragraphStyle->getStyleName());
            } elseif ($paragraphStyle->getBasedOn() != null) {
                $xmlWriter->writeElementBlock('w:basedOn', 'w:val', $paragraphStyle->getBasedOn());
            }
        }

        // w:pPr
        if (!is_null($paragraphStyle)) {
            $styleWriter = new ParagraphStyleWriter($xmlWriter, $paragraphStyle);
            $styleWriter->write();
        }

        // w:rPr
        $styleWriter = new FontStyleWriter($xmlWriter, $style);
        $styleWriter->write();

        $xmlWriter->endElement();
    }

    /**
     * Write paragraph style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $styleName
     * @param \PhpOffice\PhpWord\Style\Paragraph $style
     */
    private function writeParagraphStyle(XMLWriter $xmlWriter, $styleName, ParagraphStyle $style)
    {
        $xmlWriter->startElement('w:style');
        $xmlWriter->writeAttribute('w:type', 'paragraph');
        $xmlWriter->writeAttribute('w:customStyle', '1');
        $xmlWriter->writeAttribute('w:styleId', $styleName);
        $xmlWriter->startElement('w:name');
        $xmlWriter->writeAttribute('w:val', $styleName);
        $xmlWriter->endElement();

        // Parent style
        $basedOn = $style->getBasedOn();
        $xmlWriter->writeElementIf(!is_null($basedOn), 'w:basedOn', 'w:val', $basedOn);

        // Next paragraph style
        $next = $style->getNext();
        $xmlWriter->writeElementIf(!is_null($next), 'w:next', 'w:val', $next);

        // w:pPr
        $styleWriter = new ParagraphStyleWriter($xmlWriter, $style);
        $styleWriter->write();

        $xmlWriter->endElement();
    }

    /**
     * Write table style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $styleName
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeTableStyle(XMLWriter $xmlWriter, $styleName, TableStyle $style)
    {
        $xmlWriter->startElement('w:style');
        $xmlWriter->writeAttribute('w:type', 'table');
        $xmlWriter->writeAttribute('w:customStyle', '1');
        $xmlWriter->writeAttribute('w:styleId', $styleName);
        $xmlWriter->startElement('w:name');
        $xmlWriter->writeAttribute('w:val', $styleName);
        $xmlWriter->endElement();
        $xmlWriter->startElement('w:uiPriority');
        $xmlWriter->writeAttribute('w:val', '99');
        $xmlWriter->endElement();

        $styleWriter = new TableStyleWriter($xmlWriter, $style);
        $styleWriter->write();

        $xmlWriter->endElement(); // w:style
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/Theme.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 theme writer: word/theme/theme1.xml
 *
 * @todo Generate content dynamically
 * @since 0.10.0
 */
class Theme extends AbstractPart
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $str = '';

        $str .= '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
        $str .= '<a:theme xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" name="Office Theme">';
        $str .= '<a:themeElements>';
        $str .= $this->writeColorScheme();
        $str .= $this->writeFontScheme();
        $str .= $this->writeFormatScheme();
        $str .= '</a:themeElements>';
        $str .= '<a:objectDefaults />';
        $str .= '<a:extraClrSchemeLst />';
        $str .= '</a:theme>';

        return $str;
    }

    /**
     * Write color scheme
     *
     * @return string
     */
    private function writeColorScheme()
    {
        $str = '';

        $str .= '<a:clrScheme name="Office">';
        $str .= '<a:dk1>';
        $str .= '<a:sysClr val="windowText" lastClr="000000" />';
        $str .= '</a:dk1>';
        $str .= '<a:lt1>';
        $str .= '<a:sysClr val="window" lastClr="FFFFFF" />';
        $str .= '</a:lt1>';
        $str .= '<a:dk2>';
        $str .= '<a:srgbClr val="1F497D" />';
        $str .= '</a:dk2>';
        $str .= '<a:lt2>';
        $str .= '<a:srgbClr val="EEECE1" />';
        $str .= '</a:lt2>';
        $str .= '<a:accent1>';
        $str .= '<a:srgbClr val="4F81BD" />';
        $str .= '</a:accent1>';
        $str .= '<a:accent2>';
        $str .= '<a:srgbClr val="C0504D" />';
        $str .= '</a:accent2>';
        $str .= '<a:accent3>';
        $str .= '<a:srgbClr val="9BBB59" />';
        $str .= '</a:accent3>';
        $str .= '<a:accent4>';
        $str .= '<a:srgbClr val="8064A2" />';
        $str .= '</a:accent4>';
        $str .= '<a:accent5>';
        $str .= '<a:srgbClr val="4BACC6" />';
        $str .= '</a:accent5>';
        $str .= '<a:accent6>';
        $str .= '<a:srgbClr val="F79646" />';
        $str .= '</a:accent6>';
        $str .= '<a:hlink>';
        $str .= '<a:srgbClr val="0000FF" />';
        $str .= '</a:hlink>';
        $str .= '<a:folHlink>';
        $str .= '<a:srgbClr val="800080" />';
        $str .= '</a:folHlink>';
        $str .= '</a:clrScheme>';

        return $str;
    }

    /**
     * Write font scheme
     *
     * @return string
     */
    private function writeFontScheme()
    {
        $str = '';

        $str .= '<a:fontScheme name="Office">';

        $str .= '<a:majorFont>';
        $str .= '<a:latin typeface="Cambria" />';
        $str .= '<a:ea typeface="" />';
        $str .= '<a:cs typeface="" />';
        $str .= '<a:font script="Jpan" typeface="ＭＳ ゴシック" />';
        $str .= '<a:font script="Hang" typeface="맑은 고딕" />';
        $str .= '<a:font script="Hans" typeface="宋体" />';
        $str .= '<a:font script="Hant" typeface="新細明體" />';
        $str .= '<a:font script="Arab" typeface="Times New Roman" />';
        $str .= '<a:font script="Hebr" typeface="Times New Roman" />';
        $str .= '<a:font script="Thai" typeface="Angsana New" />';
        $str .= '<a:font script="Ethi" typeface="Nyala" />';
        $str .= '<a:font script="Beng" typeface="Vrinda" />';
        $str .= '<a:font script="Gujr" typeface="Shruti" />';
        $str .= '<a:font script="Khmr" typeface="MoolBoran" />';
        $str .= '<a:font script="Knda" typeface="Tunga" />';
        $str .= '<a:font script="Guru" typeface="Raavi" />';
        $str .= '<a:font script="Cans" typeface="Euphemia" />';
        $str .= '<a:font script="Cher" typeface="Plantagenet Cherokee" />';
        $str .= '<a:font script="Yiii" typeface="Microsoft Yi Baiti" />';
        $str .= '<a:font script="Tibt" typeface="Microsoft Himalaya" />';
        $str .= '<a:font script="Thaa" typeface="MV Boli" />';
        $str .= '<a:font script="Deva" typeface="Mangal" />';
        $str .= '<a:font script="Telu" typeface="Gautami" />';
        $str .= '<a:font script="Taml" typeface="Latha" />';
        $str .= '<a:font script="Syrc" typeface="Estrangelo Edessa" />';
        $str .= '<a:font script="Orya" typeface="Kalinga" />';
        $str .= '<a:font script="Mlym" typeface="Kartika" />';
        $str .= '<a:font script="Laoo" typeface="DokChampa" />';
        $str .= '<a:font script="Sinh" typeface="Iskoola Pota" />';
        $str .= '<a:font script="Mong" typeface="Mongolian Baiti" />';
        $str .= '<a:font script="Viet" typeface="Times New Roman" />';
        $str .= '<a:font script="Uigh" typeface="Microsoft Uighur" />';
        $str .= '</a:majorFont>';

        $str .= '<a:minorFont>';
        $str .= '<a:latin typeface="Calibri" />';
        $str .= '<a:ea typeface="" />';
        $str .= '<a:cs typeface="" />';
        $str .= '<a:font script="Jpan" typeface="ＭＳ 明朝" />';
        $str .= '<a:font script="Hang" typeface="맑은 고딕" />';
        $str .= '<a:font script="Hans" typeface="宋体" />';
        $str .= '<a:font script="Hant" typeface="新細明體" />';
        $str .= '<a:font script="Arab" typeface="Arial" />';
        $str .= '<a:font script="Hebr" typeface="Arial" />';
        $str .= '<a:font script="Thai" typeface="Cordia New" />';
        $str .= '<a:font script="Ethi" typeface="Nyala" />';
        $str .= '<a:font script="Beng" typeface="Vrinda" />';
        $str .= '<a:font script="Gujr" typeface="Shruti" />';
        $str .= '<a:font script="Khmr" typeface="DaunPenh" />';
        $str .= '<a:font script="Knda" typeface="Tunga" />';
        $str .= '<a:font script="Guru" typeface="Raavi" />';
        $str .= '<a:font script="Cans" typeface="Euphemia" />';
        $str .= '<a:font script="Cher" typeface="Plantagenet Cherokee" />';
        $str .= '<a:font script="Yiii" typeface="Microsoft Yi Baiti" />';
        $str .= '<a:font script="Tibt" typeface="Microsoft Himalaya" />';
        $str .= '<a:font script="Thaa" typeface="MV Boli" />';
        $str .= '<a:font script="Deva" typeface="Mangal" />';
        $str .= '<a:font script="Telu" typeface="Gautami" />';
        $str .= '<a:font script="Taml" typeface="Latha" />';
        $str .= '<a:font script="Syrc" typeface="Estrangelo Edessa" />';
        $str .= '<a:font script="Orya" typeface="Kalinga" />';
        $str .= '<a:font script="Mlym" typeface="Kartika" />';
        $str .= '<a:font script="Laoo" typeface="DokChampa" />';
        $str .= '<a:font script="Sinh" typeface="Iskoola Pota" />';
        $str .= '<a:font script="Mong" typeface="Mongolian Baiti" />';
        $str .= '<a:font script="Viet" typeface="Arial" />';
        $str .= '<a:font script="Uigh" typeface="Microsoft Uighur" />';
        $str .= '</a:minorFont>';

        $str .= '</a:fontScheme>';

        return $str;
    }

    /**
     * Write format scheme
     *
     * @return string
     */
    private function writeFormatScheme()
    {
        $str = '';

        $str .= '<a:fmtScheme name="Office">';
        $str .= $this->writeFormatFill();
        $str .= $this->writeFormatLine();
        $str .= $this->writeFormatEffect();
        $str .= $this->writeFormatBackground();
        $str .= '</a:fmtScheme>';

        return $str;
    }

    /**
     * Write fill format scheme
     *
     * @return string
     */
    private function writeFormatFill()
    {
        $str = '';

        $str .= '<a:fillStyleLst>';
        $str .= '<a:solidFill>';
        $str .= '<a:schemeClr val="phClr" />';
        $str .= '</a:solidFill>';
        $str .= '<a:gradFill rotWithShape="1">';
        $str .= '<a:gsLst>';
        $str .= '<a:gs pos="0">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="50000" />';
        $str .= '<a:satMod val="300000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="35000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="37000" />';
        $str .= '<a:satMod val="300000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="100000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="15000" />';
        $str .= '<a:satMod val="350000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '</a:gsLst>';
        $str .= '<a:lin ang="16200000" scaled="1" />';
        $str .= '</a:gradFill>';
        $str .= '<a:gradFill rotWithShape="1">';
        $str .= '<a:gsLst>';
        $str .= '<a:gs pos="0">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="51000" />';
        $str .= '<a:satMod val="130000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="80000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="93000" />';
        $str .= '<a:satMod val="130000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="100000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="94000" />';
        $str .= '<a:satMod val="135000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '</a:gsLst>';
        $str .= '<a:lin ang="16200000" scaled="0" />';
        $str .= '</a:gradFill>';
        $str .= '</a:fillStyleLst>';

        return $str;
    }

    /**
     * Write line format scheme
     *
     * @return string
     */
    private function writeFormatLine()
    {
        $str = '';

        $str .= '<a:lnStyleLst>';
        $str .= '<a:ln w="9525" cap="flat" cmpd="sng" algn="ctr">';
        $str .= '<a:solidFill>';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="95000" />';
        $str .= '<a:satMod val="105000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:solidFill>';
        $str .= '<a:prstDash val="solid" />';
        $str .= '</a:ln>';
        $str .= '<a:ln w="25400" cap="flat" cmpd="sng" algn="ctr">';
        $str .= '<a:solidFill>';
        $str .= '<a:schemeClr val="phClr" />';
        $str .= '</a:solidFill>';
        $str .= '<a:prstDash val="solid" />';
        $str .= '</a:ln>';
        $str .= '<a:ln w="38100" cap="flat" cmpd="sng" algn="ctr">';
        $str .= '<a:solidFill>';
        $str .= '<a:schemeClr val="phClr" />';
        $str .= '</a:solidFill>';
        $str .= '<a:prstDash val="solid" />';
        $str .= '</a:ln>';
        $str .= '</a:lnStyleLst>';

        return $str;
    }

    /**
     * Write effect format scheme
     *
     * @return string
     */
    private function writeFormatEffect()
    {
        $str = '';

        $str .= '<a:effectStyleLst>';
        $str .= '<a:effectStyle>';
        $str .= '<a:effectLst>';
        $str .= '<a:outerShdw blurRad="40000" dist="20000" dir="5400000" rotWithShape="0">';
        $str .= '<a:srgbClr val="000000">';
        $str .= '<a:alpha val="38000" />';
        $str .= '</a:srgbClr>';
        $str .= '</a:outerShdw>';
        $str .= '</a:effectLst>';
        $str .= '</a:effectStyle>';
        $str .= '<a:effectStyle>';
        $str .= '<a:effectLst>';
        $str .= '<a:outerShdw blurRad="40000" dist="23000" dir="5400000" rotWithShape="0">';
        $str .= '<a:srgbClr val="000000">';
        $str .= '<a:alpha val="35000" />';
        $str .= '</a:srgbClr>';
        $str .= '</a:outerShdw>';
        $str .= '</a:effectLst>';
        $str .= '</a:effectStyle>';
        $str .= '<a:effectStyle>';
        $str .= '<a:effectLst>';
        $str .= '<a:outerShdw blurRad="40000" dist="23000" dir="5400000" rotWithShape="0">';
        $str .= '<a:srgbClr val="000000">';
        $str .= '<a:alpha val="35000" />';
        $str .= '</a:srgbClr>';
        $str .= '</a:outerShdw>';
        $str .= '</a:effectLst>';
        $str .= '<a:scene3d>';
        $str .= '<a:camera prst="orthographicFront">';
        $str .= '<a:rot lat="0" lon="0" rev="0" />';
        $str .= '</a:camera>';
        $str .= '<a:lightRig rig="threePt" dir="t">';
        $str .= '<a:rot lat="0" lon="0" rev="1200000" />';
        $str .= '</a:lightRig>';
        $str .= '</a:scene3d>';
        $str .= '<a:sp3d>';
        $str .= '<a:bevelT w="63500" h="25400" />';
        $str .= '</a:sp3d>';
        $str .= '</a:effectStyle>';
        $str .= '</a:effectStyleLst>';

        return $str;
    }

    /**
     * Write background format scheme
     *
     * @return string
     */
    private function writeFormatBackground()
    {
        $str = '';

        $str .= '<a:bgFillStyleLst>';
        $str .= '<a:solidFill>';
        $str .= '<a:schemeClr val="phClr" />';
        $str .= '</a:solidFill>';
        $str .= '<a:gradFill rotWithShape="1">';
        $str .= '<a:gsLst>';
        $str .= '<a:gs pos="0">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="40000" />';
        $str .= '<a:satMod val="350000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="40000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="45000" />';
        $str .= '<a:shade val="99000" />';
        $str .= '<a:satMod val="350000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="100000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="20000" />';
        $str .= '<a:satMod val="255000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '</a:gsLst>';
        $str .= '<a:path path="circle">';
        $str .= '<a:fillToRect l="50000" t="-80000" r="50000" b="180000" />';
        $str .= '</a:path>';
        $str .= '</a:gradFill>';
        $str .= '<a:gradFill rotWithShape="1">';
        $str .= '<a:gsLst>';
        $str .= '<a:gs pos="0">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:tint val="80000" />';
        $str .= '<a:satMod val="300000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '<a:gs pos="100000">';
        $str .= '<a:schemeClr val="phClr">';
        $str .= '<a:shade val="30000" />';
        $str .= '<a:satMod val="200000" />';
        $str .= '</a:schemeClr>';
        $str .= '</a:gs>';
        $str .= '</a:gsLst>';
        $str .= '<a:path path="circle">';
        $str .= '<a:fillToRect l="50000" t="50000" r="50000" b="50000" />';
        $str .= '</a:path>';
        $str .= '</a:gradFill>';
        $str .= '</a:bgFillStyleLst>';

        return $str;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Part/WebSettings.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Part;

/**
 * Word2007 web settings part writer: word/webSettings.xml
 */
class WebSettings extends Settings
{
    /**
     * Write part
     *
     * @return string
     */
    public function write()
    {
        $settings = array(
            'w:optimizeForBrowser' => '',
        );

        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startDocument('1.0', 'UTF-8', 'yes');
        $xmlWriter->startElement('w:webSettings');
        $xmlWriter->writeAttribute('xmlns:r', 'http://schemas.openxmlformats.org/officeDocument/2006/relationships');
        $xmlWriter->writeAttribute('xmlns:w', 'http://schemas.openxmlformats.org/wordprocessingml/2006/main');

        foreach ($settings as $settingKey => $settingValue) {
            $this->writeSetting($xmlWriter, $settingKey, $settingValue);
        }

        $xmlWriter->endElement(); // w:settings

        return $xmlWriter->getData();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/AbstractStyle.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Settings;
use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Style writer
 *
 * @since 0.10.0
 */
abstract class AbstractStyle
{
    /**
     * XML writer
     *
     * @var \PhpOffice\PhpWord\Shared\XMLWriter
     */
    private $xmlWriter;

    /**
     * Style; set protected for a while
     *
     * @var string|\PhpOffice\PhpWord\Style\AbstractStyle
     */
    protected $style;

    /**
     * Write style
     */
    abstract public function write();

    /**
     * Create new instance.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string|\PhpOffice\PhpWord\Style\AbstractStyle $style
     */
    public function __construct(XMLWriter $xmlWriter, $style = null)
    {
        $this->xmlWriter = $xmlWriter;
        $this->style = $style;
    }

    /**
     * Get XML Writer
     *
     * @return \PhpOffice\PhpWord\Shared\XMLWriter
     */
    protected function getXmlWriter()
    {
        return $this->xmlWriter;
    }

    /**
     * Get Style
     *
     * @return string|\PhpOffice\PhpWord\Style\AbstractStyle
     */
    protected function getStyle()
    {
        return $this->style;
    }

    /**
     * Convert twip value
     *
     * @param int|float $value
     * @param int $default (int|float)
     * @return int|float
     */
    protected function convertTwip($value, $default = 0)
    {
        $factors = array(
            Settings::UNIT_CM    => 567,
            Settings::UNIT_MM    => 56.7,
            Settings::UNIT_INCH  => 1440,
            Settings::UNIT_POINT => 20,
            Settings::UNIT_PICA  => 240,
        );
        $unit = Settings::getMeasurementUnit();
        $factor = 1;
        if (array_key_exists($unit, $factors) && $value != $default) {
            $factor = $factors[$unit];
        }

        return $value * $factor;
    }

    /**
     * Write child style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $name
     * @param mixed $value
     */
    protected function writeChildStyle(XMLWriter $xmlWriter, $name, $value)
    {
        if ($value !== null) {
            $class = 'PhpOffice\\PhpWord\\Writer\\Word2007\\Style\\' . $name;

            /** @var \PhpOffice\PhpWord\Writer\Word2007\Style\AbstractStyle $writer */
            $writer = new $class($xmlWriter, $value);
            $writer->write();
        }
    }

    /**
     * Writes boolean as 0 or 1
     *
     * @param bool $value
     * @return null|string
     */
    protected function writeOnOf($value = null)
    {
        if ($value === null) {
            return null;
        }

        return $value ? '1' : '0';
    }

    /**
     * Assemble style array into style string
     *
     * @param array $styles
     * @return string
     */
    protected function assembleStyle($styles = array())
    {
        $style = '';
        foreach ($styles as $key => $value) {
            if (!is_null($value) && $value != '') {
                $style .= "{$key}:{$value}; ";
            }
        }

        return trim($style);
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Cell.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Style\Cell as CellStyle;

/**
 * Cell style writer
 *
 * @since 0.10.0
 */
class Cell extends AbstractStyle
{
    /**
     * @var int Cell width
     */
    private $width;

    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof CellStyle) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:tcPr');

        // Width
        if (!is_null($this->width) || !is_null($style->getWidth())) {
            $width = is_null($this->width) ? $style->getWidth() : $this->width;

            $xmlWriter->startElement('w:tcW');
            $xmlWriter->writeAttribute('w:w', $width);
            $xmlWriter->writeAttribute('w:type', $style->getUnit());
            $xmlWriter->endElement(); // w:tcW
        }

        // Text direction
        $textDir = $style->getTextDirection();
        $xmlWriter->writeElementIf(!is_null($textDir), 'w:textDirection', 'w:val', $textDir);

        // Vertical alignment
        $vAlign = $style->getVAlign();
        $xmlWriter->writeElementIf(!is_null($vAlign), 'w:vAlign', 'w:val', $vAlign);

        // Border
        if ($style->hasBorder()) {
            $xmlWriter->startElement('w:tcBorders');

            $styleWriter = new MarginBorder($xmlWriter);
            $styleWriter->setSizes($style->getBorderSize());
            $styleWriter->setColors($style->getBorderColor());
            $styleWriter->setStyles($style->getBorderStyle());
            $styleWriter->setAttributes(array('defaultColor' => CellStyle::DEFAULT_BORDER_COLOR));
            $styleWriter->write();

            $xmlWriter->endElement();
        }

        // Shading
        $shading = $style->getShading();
        if (!is_null($shading)) {
            $styleWriter = new Shading($xmlWriter, $shading);
            $styleWriter->write();
        }

        // Colspan & rowspan
        $gridSpan = $style->getGridSpan();
        $vMerge = $style->getVMerge();
        $xmlWriter->writeElementIf(!is_null($gridSpan), 'w:gridSpan', 'w:val', $gridSpan);
        $xmlWriter->writeElementIf(!is_null($vMerge), 'w:vMerge', 'w:val', $vMerge);

        $xmlWriter->endElement(); // w:tcPr
    }

    /**
     * Set width.
     *
     * @param int $value
     */
    public function setWidth($value = null)
    {
        $this->width = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Extrusion.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * 3D extrusion style writer
 *
 * @since 0.12.0
 */
class Extrusion extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Extrusion) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('o:extrusion');
        $xmlWriter->writeAttribute('on', 't');
        $xmlWriter->writeAttributeIf($style->getType() !== null, 'type', $style->getType());
        $xmlWriter->writeAttributeIf($style->getColor() !== null, 'color', $style->getColor());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Fill.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Fill style writer
 *
 * @since 0.12.0
 */
class Fill extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Fill) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->writeAttribute('on', 't');
        $xmlWriter->writeAttributeIf($style->getColor() !== null, 'fillcolor', $style->getColor());
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Font.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Font style writer
 *
 * @since 0.10.0
 */
class Font extends AbstractStyle
{
    /**
     * Is inline in element
     *
     * @var bool
     */
    private $isInline = false;

    /**
     * Write style.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $isStyleName = $this->isInline && !is_null($this->style) && is_string($this->style);
        if ($isStyleName) {
            $xmlWriter->startElement('w:rPr');
            $xmlWriter->startElement('w:rStyle');
            $xmlWriter->writeAttribute('w:val', $this->style);
            $xmlWriter->endElement();
            $style = \PhpOffice\PhpWord\Style::getStyle($this->style);
            if ($style instanceof \PhpOffice\PhpWord\Style\Font) {
                $xmlWriter->writeElementIf($style->isRTL(), 'w:rtl');
            }
            $xmlWriter->endElement();
        } else {
            $this->writeStyle();
        }
    }

    /**
     * Write full style.
     */
    private function writeStyle()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Font) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:rPr');

        // Style name
        if ($this->isInline === true) {
            $styleName = $style->getStyleName();
            $xmlWriter->writeElementIf($styleName !== null, 'w:rStyle', 'w:val', $styleName);
        }

        // Font name/family
        $font = $style->getName();
        $hint = $style->getHint();
        if ($font !== null) {
            $xmlWriter->startElement('w:rFonts');
            $xmlWriter->writeAttribute('w:ascii', $font);
            $xmlWriter->writeAttribute('w:hAnsi', $font);
            $xmlWriter->writeAttribute('w:eastAsia', $font);
            $xmlWriter->writeAttribute('w:cs', $font);
            $xmlWriter->writeAttributeIf($hint !== null, 'w:hint', $hint);
            $xmlWriter->endElement();
        }

        //Language
        $language = $style->getLang();
        if ($language != null && ($language->getLatin() !== null || $language->getEastAsia() !== null || $language->getBidirectional() !== null)) {
            $xmlWriter->startElement('w:lang');
            $xmlWriter->writeAttributeIf($language->getLatin() !== null, 'w:val', $language->getLatin());
            $xmlWriter->writeAttributeIf($language->getEastAsia() !== null, 'w:eastAsia', $language->getEastAsia());
            $xmlWriter->writeAttributeIf($language->getBidirectional() !== null, 'w:bidi', $language->getBidirectional());
            //if bidi is not set but we are writing RTL, write the latin language in the bidi tag
            if ($style->isRTL() && $language->getBidirectional() === null && $language->getLatin() !== null) {
                $xmlWriter->writeAttribute('w:bidi', $language->getLatin());
            }
            $xmlWriter->endElement();
        }

        // Color
        $color = $style->getColor();
        $xmlWriter->writeElementIf($color !== null, 'w:color', 'w:val', $color);

        // Size
        $size = $style->getSize();
        $xmlWriter->writeElementIf($size !== null, 'w:sz', 'w:val', $size * 2);
        $xmlWriter->writeElementIf($size !== null, 'w:szCs', 'w:val', $size * 2);

        // Bold, italic
        $xmlWriter->writeElementIf($style->isBold() !== null, 'w:b', 'w:val', $this->writeOnOf($style->isBold()));
        $xmlWriter->writeElementIf($style->isBold() !== null, 'w:bCs', 'w:val', $this->writeOnOf($style->isBold()));
        $xmlWriter->writeElementIf($style->isItalic() !== null, 'w:i', 'w:val', $this->writeOnOf($style->isItalic()));
        $xmlWriter->writeElementIf($style->isItalic() !== null, 'w:iCs', 'w:val', $this->writeOnOf($style->isItalic()));

        // Strikethrough, double strikethrough
        $xmlWriter->writeElementIf($style->isStrikethrough() !== null, 'w:strike', 'w:val', $this->writeOnOf($style->isStrikethrough()));
        $xmlWriter->writeElementIf($style->isDoubleStrikethrough() !== null, 'w:dstrike', 'w:val', $this->writeOnOf($style->isDoubleStrikethrough()));

        // Small caps, all caps
        $xmlWriter->writeElementIf($style->isSmallCaps() !== null, 'w:smallCaps', 'w:val', $this->writeOnOf($style->isSmallCaps()));
        $xmlWriter->writeElementIf($style->isAllCaps() !== null, 'w:caps', 'w:val', $this->writeOnOf($style->isAllCaps()));

        //Hidden text
        $xmlWriter->writeElementIf($style->isHidden(), 'w:vanish', 'w:val', $this->writeOnOf($style->isHidden()));

        // Underline
        $xmlWriter->writeElementIf($style->getUnderline() != 'none', 'w:u', 'w:val', $style->getUnderline());

        // Foreground-Color
        $xmlWriter->writeElementIf($style->getFgColor() !== null, 'w:highlight', 'w:val', $style->getFgColor());

        // Superscript/subscript
        $xmlWriter->writeElementIf($style->isSuperScript(), 'w:vertAlign', 'w:val', 'superscript');
        $xmlWriter->writeElementIf($style->isSubScript(), 'w:vertAlign', 'w:val', 'subscript');

        // Spacing
        $xmlWriter->writeElementIf($style->getScale() !== null, 'w:w', 'w:val', $style->getScale());
        $xmlWriter->writeElementIf($style->getSpacing() !== null, 'w:spacing', 'w:val', $style->getSpacing());
        $xmlWriter->writeElementIf($style->getKerning() !== null, 'w:kern', 'w:val', $style->getKerning() * 2);

        // noProof
        $xmlWriter->writeElementIf($style->isNoProof() !== null, 'w:noProof', 'w:val', $this->writeOnOf($style->isNoProof()));

        // Background-Color
        $shading = $style->getShading();
        if (!is_null($shading)) {
            $styleWriter = new Shading($xmlWriter, $shading);
            $styleWriter->write();
        }

        // RTL
        if ($this->isInline === true) {
            $styleName = $style->getStyleName();
            $xmlWriter->writeElementIf($styleName === null && $style->isRTL(), 'w:rtl');
        }

        // Position
        $xmlWriter->writeElementIf($style->getPosition() !== null, 'w:position', 'w:val', $style->getPosition());

        $xmlWriter->endElement();
    }

    /**
     * Set is inline.
     *
     * @param bool $value
     */
    public function setIsInline($value)
    {
        $this->isInline = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Frame.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style\Frame as FrameStyle;
use PhpOffice\PhpWord\Writer\Word2007\Element\ParagraphAlignment;

/**
 * Frame style writer
 *
 * @since 0.12.0
 */
class Frame extends AbstractStyle
{
    const PHP_32BIT_INT_MAX = 2147483647;

    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof FrameStyle) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $maxZIndex = min(PHP_INT_MAX, self::PHP_32BIT_INT_MAX);
        $zIndices = array(FrameStyle::WRAP_INFRONT => $maxZIndex, FrameStyle::WRAP_BEHIND => -$maxZIndex);

        $properties = array(
            'width'              => 'width',
            'height'             => 'height',
            'left'               => 'margin-left',
            'top'                => 'margin-top',
            'wrapDistanceTop'    => 'mso-wrap-distance-top',
            'wrapDistanceBottom' => 'mso-wrap-distance-bottom',
            'wrapDistanceLeft'   => 'mso-wrap-distance-left',
            'wrapDistanceRight'  => 'mso-wrap-distance-right',
        );
        $sizeStyles = $this->getStyles($style, $properties, $style->getUnit());

        $properties = array(
            'pos'       => 'position',
            'hPos'      => 'mso-position-horizontal',
            'vPos'      => 'mso-position-vertical',
            'hPosRelTo' => 'mso-position-horizontal-relative',
            'vPosRelTo' => 'mso-position-vertical-relative',
        );
        $posStyles = $this->getStyles($style, $properties);

        $styles = array_merge($sizeStyles, $posStyles);

        // zIndex for infront & behind wrap
        $wrap = $style->getWrap();
        if ($wrap !== null && isset($zIndices[$wrap])) {
            $styles['z-index'] = $zIndices[$wrap];
            $wrap = null;
        }

        // Style attribute
        $xmlWriter->writeAttribute('style', $this->assembleStyle($styles));

        $this->writeWrap($xmlWriter, $style, $wrap);
    }

    /**
     * Write alignment.
     */
    public function writeAlignment()
    {
        $style = $this->getStyle();
        if (!$style instanceof FrameStyle) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('w:pPr');

        if ('' !== $style->getAlignment()) {
            $paragraphAlignment = new ParagraphAlignment($style->getAlignment());
            $xmlWriter->startElement($paragraphAlignment->getName());
            foreach ($paragraphAlignment->getAttributes() as $attributeName => $attributeValue) {
                $xmlWriter->writeAttribute($attributeName, $attributeValue);
            }
            $xmlWriter->endElement();
        }

        $xmlWriter->endElement();
    }

    /**
     * Write wrap.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Frame $style
     * @param string $wrap
     */
    private function writeWrap(XMLWriter $xmlWriter, FrameStyle $style, $wrap)
    {
        if ($wrap !== null) {
            $xmlWriter->startElement('w10:wrap');
            $xmlWriter->writeAttribute('type', $wrap);

            $relativePositions = array(
                FrameStyle::POS_RELTO_MARGIN  => 'margin',
                FrameStyle::POS_RELTO_PAGE    => 'page',
                FrameStyle::POS_RELTO_TMARGIN => 'margin',
                FrameStyle::POS_RELTO_BMARGIN => 'page',
                FrameStyle::POS_RELTO_LMARGIN => 'margin',
                FrameStyle::POS_RELTO_RMARGIN => 'page',
            );
            $pos = $style->getPos();
            $hPos = $style->getHPosRelTo();
            $vPos = $style->getVPosRelTo();

            if ($pos == FrameStyle::POS_ABSOLUTE) {
                $xmlWriter->writeAttribute('anchorx', 'page');
                $xmlWriter->writeAttribute('anchory', 'page');
            } elseif ($pos == FrameStyle::POS_RELATIVE) {
                if (isset($relativePositions[$hPos])) {
                    $xmlWriter->writeAttribute('anchorx', $relativePositions[$hPos]);
                }
                if (isset($relativePositions[$vPos])) {
                    $xmlWriter->writeAttribute('anchory', $relativePositions[$vPos]);
                }
            }

            $xmlWriter->endElement(); // w10:wrap
        }
    }

    /**
     * Get style values in associative array
     *
     * @param \PhpOffice\PhpWord\Style\Frame $style
     * @param array $properties
     * @param string $suffix
     * @return array
     */
    private function getStyles(FrameStyle $style, $properties, $suffix = '')
    {
        $styles = array();

        foreach ($properties as $key => $property) {
            $method = "get{$key}";
            $value = $style->$method();
            if ($value !== null) {
                $styles[$property] = $style->$method() . $suffix;
            }
        }

        return $styles;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Image.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Image style writer
 *
 * @since 0.10.0
 */
class Image extends Frame
{
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Indentation.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Paragraph indentation style writer
 *
 * @since 0.10.0
 */
class Indentation extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Indentation) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:ind');

        $xmlWriter->writeAttribute('w:left', $this->convertTwip($style->getLeft()));
        $xmlWriter->writeAttribute('w:right', $this->convertTwip($style->getRight()));

        $firstLine = $style->getFirstLine();
        $xmlWriter->writeAttributeIf(!is_null($firstLine), 'w:firstLine', $this->convertTwip($firstLine));

        $hanging = $style->getHanging();
        $xmlWriter->writeAttributeIf(!is_null($hanging), 'w:hanging', $this->convertTwip($hanging));

        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Line.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Style\Line as LineStyle;

/**
 * Line style writer
 */
class Line extends Frame
{
    /**
     * Write Line stroke.
     * @todo Merge with `Stroke` style
     */
    public function writeStroke()
    {
        $xmlWriter = $this->getXmlWriter();
        $style = $this->getStyle();
        if (!$style instanceof LineStyle) {
            return;
        }

        $dash = $style->getDash();
        $dashStyles = array(
            LineStyle::DASH_STYLE_DASH              => 'dash',
            LineStyle::DASH_STYLE_ROUND_DOT         => '1 1',
            LineStyle::DASH_STYLE_SQUARE_DOT        => '1 1',
            LineStyle::DASH_STYLE_DASH_DOT          => 'dashDot',
            LineStyle::DASH_STYLE_LONG_DASH         => 'longDash',
            LineStyle::DASH_STYLE_LONG_DASH_DOT     => 'longDashDot',
            LineStyle::DASH_STYLE_LONG_DASH_DOT_DOT => 'longDashDotDot',
        );

        $xmlWriter->startElement('v:stroke');

        $xmlWriter->writeAttributeIf($style->getWeight() !== null, 'weight', $style->getWeight() . 'pt');
        $xmlWriter->writeAttributeIf($style->getColor() !== null, 'color', $style->getColor());
        $xmlWriter->writeAttributeIf($style->getBeginArrow() !== null, 'startarrow', $style->getBeginArrow());
        $xmlWriter->writeAttributeIf($style->getEndArrow() !== null, 'endarrow', $style->getEndArrow());

        if ($dash !== null) {
            if (isset($dashStyles[$dash])) {
                $xmlWriter->writeAttribute('dashstyle', $dashStyles[$dash]);
            }
            if ($dash == LineStyle::DASH_STYLE_ROUND_DOT) {
                $xmlWriter->writeAttribute('endcap', 'round');
            }
        }

        $xmlWriter->endElement(); //v:stroke
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/LineNumbering.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Line numbering style writer
 *
 * @since 0.10.0
 */
class LineNumbering extends AbstractStyle
{
    /**
     * Write style.
     * The w:start seems to be zero based so we have to decrement by one
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\LineNumbering) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:lnNumType');
        $xmlWriter->writeAttribute('w:start', $style->getStart() - 1);
        $xmlWriter->writeAttribute('w:countBy', $style->getIncrement());
        $xmlWriter->writeAttribute('w:distance', $style->getDistance());
        $xmlWriter->writeAttribute('w:restart', $style->getRestart());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/MarginBorder.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Shared\XMLWriter;

/**
 * Margin border style writer
 *
 * @since 0.10.0
 */
class MarginBorder extends AbstractStyle
{
    /**
     * Sizes
     *
     * @var int[]
     */
    private $sizes = array();

    /**
     * Colors
     *
     * @var string[]
     */
    private $colors = array();

    /**
     * Border styles
     *
     * @var string[]
     */
    private $styles = array();

    /**
     * Other attributes
     *
     * @var array
     */
    private $attributes = array();

    /**
     * Write style.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $sides = array('top', 'left', 'right', 'bottom', 'insideH', 'insideV');

        foreach ($this->sizes as $i => $size) {
            if ($size !== null) {
                $color = null;
                if (isset($this->colors[$i])) {
                    $color = $this->colors[$i];
                }
                $style = isset($this->styles[$i]) ? $this->styles[$i] : 'single';
                $this->writeSide($xmlWriter, $sides[$i], $this->sizes[$i], $color, $style);
            }
        }
    }

    /**
     * Write side.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $side
     * @param int $width
     * @param string $color
     * @param string $borderStyle
     */
    private function writeSide(XMLWriter $xmlWriter, $side, $width, $color = null, $borderStyle = 'solid')
    {
        $xmlWriter->startElement('w:' . $side);
        if (!empty($this->colors)) {
            if ($color === null && !empty($this->attributes)) {
                if (isset($this->attributes['defaultColor'])) {
                    $color = $this->attributes['defaultColor'];
                }
            }
            $xmlWriter->writeAttribute('w:val', $borderStyle);
            $xmlWriter->writeAttribute('w:sz', $width);
            $xmlWriter->writeAttributeIf($color != null, 'w:color', $color);
            if (!empty($this->attributes)) {
                if (isset($this->attributes['space'])) {
                    $xmlWriter->writeAttribute('w:space', $this->attributes['space']);
                }
            }
        } else {
            $xmlWriter->writeAttribute('w:w', $width);
            $xmlWriter->writeAttribute('w:type', 'dxa');
        }
        $xmlWriter->endElement();
    }

    /**
     * Set sizes.
     *
     * @param int[] $value
     */
    public function setSizes($value)
    {
        $this->sizes = $value;
    }

    /**
     * Set colors.
     *
     * @param string[] $value
     */
    public function setColors($value)
    {
        $this->colors = $value;
    }

    /**
     * Set border styles.
     *
     * @param string[] $value
     */
    public function setStyles($value)
    {
        $this->styles = $value;
    }

    /**
     * Set attributes.
     *
     * @param array $value
     */
    public function setAttributes($value)
    {
        $this->attributes = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Outline.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Outline style writer
 *
 * @since 0.12.0
 */
class Outline extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Outline) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('v:stroke');
        $xmlWriter->writeAttribute('on', 't');
        $xmlWriter->writeAttributeIf($style->getColor() !== null, 'color', $style->getColor());
        $xmlWriter->writeAttributeIf($style->getWeight() !== null, 'weight', $style->getWeight() . $style->getUnit());
        $xmlWriter->writeAttributeIf($style->getDash() !== null, 'dashstyle', $style->getDash());
        $xmlWriter->writeAttributeIf($style->getLine() !== null, 'linestyle', $style->getLine());
        $xmlWriter->writeAttributeIf($style->getEndCap() !== null, 'endcap', $style->getEndCap());
        $xmlWriter->writeAttributeIf($style->getStartArrow() !== null, 'startarrow', $style->getStartArrow());
        $xmlWriter->writeAttributeIf($style->getEndArrow() !== null, 'endarrow', $style->getEndArrow());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Paragraph.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\Style;
use PhpOffice\PhpWord\Style\Paragraph as ParagraphStyle;
use PhpOffice\PhpWord\Writer\Word2007\Element\ParagraphAlignment;

/**
 * Paragraph style writer
 *
 * @since 0.10.0
 */
class Paragraph extends AbstractStyle
{
    /**
     * Without w:pPr
     *
     * @var bool
     */
    private $withoutPPR = false;

    /**
     * Is inline in element
     *
     * @var bool
     */
    private $isInline = false;

    /**
     * Write style.
     */
    public function write()
    {
        $xmlWriter = $this->getXmlWriter();

        $isStyleName = $this->isInline && !is_null($this->style) && is_string($this->style);
        if ($isStyleName) {
            if (!$this->withoutPPR) {
                $xmlWriter->startElement('w:pPr');
            }
            $xmlWriter->startElement('w:pStyle');
            $xmlWriter->writeAttribute('w:val', $this->style);
            $xmlWriter->endElement();
            if (!$this->withoutPPR) {
                $xmlWriter->endElement();
            }
        } else {
            $this->writeStyle();
        }
    }

    /**
     * Write full style.
     */
    private function writeStyle()
    {
        $style = $this->getStyle();
        if (!$style instanceof ParagraphStyle) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();
        $styles = $style->getStyleValues();

        if (!$this->withoutPPR) {
            $xmlWriter->startElement('w:pPr');
        }

        // Style name
        if ($this->isInline === true) {
            $xmlWriter->writeElementIf($styles['name'] !== null, 'w:pStyle', 'w:val', $styles['name']);
        }

        // Pagination
        $xmlWriter->writeElementIf($styles['pagination']['widowControl'] === false, 'w:widowControl', 'w:val', '0');
        $xmlWriter->writeElementIf($styles['pagination']['keepNext'] === true, 'w:keepNext', 'w:val', '1');
        $xmlWriter->writeElementIf($styles['pagination']['keepLines'] === true, 'w:keepLines', 'w:val', '1');
        $xmlWriter->writeElementIf($styles['pagination']['pageBreak'] === true, 'w:pageBreakBefore', 'w:val', '1');

        // Paragraph alignment
        if ('' !== $styles['alignment']) {
            $paragraphAlignment = new ParagraphAlignment($styles['alignment']);
            $xmlWriter->startElement($paragraphAlignment->getName());
            foreach ($paragraphAlignment->getAttributes() as $attributeName => $attributeValue) {
                $xmlWriter->writeAttribute($attributeName, $attributeValue);
            }
            $xmlWriter->endElement();
        }

        //Right to left
        $xmlWriter->writeElementIf($styles['bidi'] === true, 'w:bidi');

        //Paragraph contextualSpacing
        $xmlWriter->writeElementIf($styles['contextualSpacing'] === true, 'w:contextualSpacing');

        //Paragraph textAlignment
        $xmlWriter->writeElementIf($styles['textAlignment'] !== null, 'w:textAlignment', 'w:val', $styles['textAlignment']);

        // Hyphenation
        $xmlWriter->writeElementIf($styles['suppressAutoHyphens'] === true, 'w:suppressAutoHyphens');

        // Child style: alignment, indentation, spacing, and shading
        $this->writeChildStyle($xmlWriter, 'Indentation', $styles['indentation']);
        $this->writeChildStyle($xmlWriter, 'Spacing', $styles['spacing']);
        $this->writeChildStyle($xmlWriter, 'Shading', $styles['shading']);

        // Tabs
        $this->writeTabs($xmlWriter, $styles['tabs']);

        // Numbering
        $this->writeNumbering($xmlWriter, $styles['numbering']);

        // Border
        if ($style->hasBorder()) {
            $xmlWriter->startElement('w:pBdr');

            $styleWriter = new MarginBorder($xmlWriter);
            $styleWriter->setSizes($style->getBorderSize());
            $styleWriter->setStyles($style->getBorderStyle());
            $styleWriter->setColors($style->getBorderColor());
            $styleWriter->write();

            $xmlWriter->endElement();
        }

        if (!$this->withoutPPR) {
            $xmlWriter->endElement(); // w:pPr
        }
    }

    /**
     * Write tabs.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Tab[] $tabs
     */
    private function writeTabs(XMLWriter $xmlWriter, $tabs)
    {
        if (!empty($tabs)) {
            $xmlWriter->startElement('w:tabs');
            foreach ($tabs as $tab) {
                $styleWriter = new Tab($xmlWriter, $tab);
                $styleWriter->write();
            }
            $xmlWriter->endElement();
        }
    }

    /**
     * Write numbering.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param array $numbering
     */
    private function writeNumbering(XMLWriter $xmlWriter, $numbering)
    {
        $numStyle = $numbering['style'];
        $numLevel = $numbering['level'];

        /** @var \PhpOffice\PhpWord\Style\Numbering $numbering */
        $numbering = Style::getStyle($numStyle);
        if ($numStyle !== null && $numbering !== null) {
            $xmlWriter->startElement('w:numPr');
            $xmlWriter->startElement('w:numId');
            $xmlWriter->writeAttribute('w:val', $numbering->getIndex());
            $xmlWriter->endElement(); // w:numId
            $xmlWriter->startElement('w:ilvl');
            $xmlWriter->writeAttribute('w:val', $numLevel);
            $xmlWriter->endElement(); // w:ilvl
            $xmlWriter->endElement(); // w:numPr

            $xmlWriter->startElement('w:outlineLvl');
            $xmlWriter->writeAttribute('w:val', $numLevel);
            $xmlWriter->endElement(); // w:outlineLvl
        }
    }

    /**
     * Set without w:pPr.
     *
     * @param bool $value
     */
    public function setWithoutPPR($value)
    {
        $this->withoutPPR = $value;
    }

    /**
     * Set is inline.
     *
     * @param bool $value
     */
    public function setIsInline($value)
    {
        $this->isInline = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Row.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Row style writer
 *
 * @since 0.11.0
 */
class Row extends AbstractStyle
{
    /**
     * @var int Row height
     */
    private $height;

    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Row) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();
        $xmlWriter->startElement('w:trPr');

        if ($this->height !== null) {
            $xmlWriter->startElement('w:trHeight');
            $xmlWriter->writeAttribute('w:val', $this->height);
            $xmlWriter->writeAttribute('w:hRule', ($style->isExactHeight() ? 'exact' : 'atLeast'));
            $xmlWriter->endElement();
        }
        $xmlWriter->writeElementIf($style->isTblHeader(), 'w:tblHeader', 'w:val', '1');
        $xmlWriter->writeElementIf($style->isCantSplit(), 'w:cantSplit', 'w:val', '1');

        $xmlWriter->endElement(); // w:trPr
    }

    /**
     * Set height.
     *
     * @param int $value
     */
    public function setHeight($value = null)
    {
        $this->height = $value;
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Section.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Style\Section as SectionStyle;

/**
 * Section style writer
 *
 * @since 0.10.0
 */
class Section extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof SectionStyle) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        // Break type
        $breakType = $style->getBreakType();
        $xmlWriter->writeElementIf(!is_null($breakType), 'w:type', 'w:val', $breakType);

        // Page size & orientation
        $xmlWriter->startElement('w:pgSz');
        $xmlWriter->writeAttribute('w:orient', $style->getOrientation());
        $xmlWriter->writeAttribute('w:w', $style->getPageSizeW());
        $xmlWriter->writeAttribute('w:h', $style->getPageSizeH());
        $xmlWriter->endElement(); // w:pgSz

        // Vertical alignment
        $vAlign = $style->getVAlign();
        $xmlWriter->writeElementIf(!is_null($vAlign), 'w:vAlign', 'w:val', $vAlign);

        // Margins
        $margins = array(
            'w:top'    => array('getMarginTop', SectionStyle::DEFAULT_MARGIN),
            'w:right'  => array('getMarginRight', SectionStyle::DEFAULT_MARGIN),
            'w:bottom' => array('getMarginBottom', SectionStyle::DEFAULT_MARGIN),
            'w:left'   => array('getMarginLeft', SectionStyle::DEFAULT_MARGIN),
            'w:header' => array('getHeaderHeight', SectionStyle::DEFAULT_HEADER_HEIGHT),
            'w:footer' => array('getFooterHeight', SectionStyle::DEFAULT_FOOTER_HEIGHT),
            'w:gutter' => array('getGutter', SectionStyle::DEFAULT_GUTTER),
        );
        $xmlWriter->startElement('w:pgMar');
        foreach ($margins as $attribute => $value) {
            list($method, $default) = $value;
            $xmlWriter->writeAttribute($attribute, $this->convertTwip($style->$method(), $default));
        }
        $xmlWriter->endElement();

        // Borders
        if ($style->hasBorder()) {
            $xmlWriter->startElement('w:pgBorders');
            $xmlWriter->writeAttribute('w:offsetFrom', 'page');

            $styleWriter = new MarginBorder($xmlWriter);
            $styleWriter->setSizes($style->getBorderSize());
            $styleWriter->setColors($style->getBorderColor());
            $styleWriter->setAttributes(array('space' => '24'));
            $styleWriter->write();

            $xmlWriter->endElement();
        }

        // Columns
        $colsSpace = $style->getColsSpace();
        $xmlWriter->startElement('w:cols');
        $xmlWriter->writeAttribute('w:num', $style->getColsNum());
        $xmlWriter->writeAttribute('w:space', $this->convertTwip($colsSpace, SectionStyle::DEFAULT_COLUMN_SPACING));
        $xmlWriter->endElement();

        // Page numbering start
        $pageNum = $style->getPageNumberingStart();
        $xmlWriter->writeElementIf(!is_null($pageNum), 'w:pgNumType', 'w:start', $pageNum);

        // Line numbering
        $styleWriter = new LineNumbering($xmlWriter, $style->getLineNumbering());
        $styleWriter->write();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Shading.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Shading style writer
 *
 * @since 0.10.0
 */
class Shading extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Shading) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:shd');
        $xmlWriter->writeAttributeIf(!is_null($style->getPattern()), 'w:val', $style->getPattern());
        $xmlWriter->writeAttributeIf(!is_null($style->getColor()), 'w:color', $style->getColor());
        $xmlWriter->writeAttributeIf(!is_null($style->getFill()), 'w:fill', $style->getFill());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Shadow.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Shadow style writer
 *
 * @since 0.12.0
 */
class Shadow extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Shadow) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('v:shadow');
        $xmlWriter->writeAttribute('on', 't');
        $xmlWriter->writeAttributeIf($style->getColor() !== null, 'color', $style->getColor());
        $xmlWriter->writeAttributeIf($style->getOffset() !== null, 'offset', $style->getOffset());
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Shape.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Shape style writer
 *
 * @since 0.12.0
 */
class Shape extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Shape) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();

        $childStyles = array('Frame', 'Fill', 'Outline', 'Shadow', 'Extrusion');
        foreach ($childStyles as $childStyle) {
            $method = "get{$childStyle}";
            $this->writeChildStyle($xmlWriter, $childStyle, $style->$method());
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Spacing.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Spacing between lines and above/below paragraph style writer
 *
 * @since 0.10.0
 */
class Spacing extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Spacing) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:spacing');

        $before = $style->getBefore();
        $xmlWriter->writeAttributeIf(!is_null($before), 'w:before', $this->convertTwip($before));

        $after = $style->getAfter();
        $xmlWriter->writeAttributeIf(!is_null($after), 'w:after', $this->convertTwip($after));

        $line = $style->getLine();
        //if linerule is auto, the spacing is supposed to include the height of the line itself, which is 240 twips
        if (null !== $line && 'auto' === $style->getLineRule()) {
            $line += \PhpOffice\PhpWord\Style\Paragraph::LINE_HEIGHT;
        }
        $xmlWriter->writeAttributeIf(!is_null($line), 'w:line', $line);

        $xmlWriter->writeAttributeIf(!is_null($line), 'w:lineRule', $style->getLineRule());

        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Tab.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * Line numbering style writer
 *
 * @since 0.10.0
 */
class Tab extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\Tab) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('w:tab');
        $xmlWriter->writeAttribute('w:val', $style->getType());
        $xmlWriter->writeAttribute('w:leader', $style->getLeader());
        $xmlWriter->writeAttribute('w:pos', $this->convertTwip($style->getPosition()));
        $xmlWriter->endElement();
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/Table.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Shared\XMLWriter;
use PhpOffice\PhpWord\SimpleType\TblWidth;
use PhpOffice\PhpWord\Style\Table as TableStyle;
use PhpOffice\PhpWord\Writer\Word2007\Element\TableAlignment;

/**
 * Table style writer
 *
 * @since 0.10.0
 */
class Table extends AbstractStyle
{
    /**
     * @var int Table width
     */
    private $width;

    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        $xmlWriter = $this->getXmlWriter();

        if ($style instanceof TableStyle) {
            $this->writeStyle($xmlWriter, $style);
        } elseif (is_string($style)) {
            $xmlWriter->startElement('w:tblPr');
            $xmlWriter->startElement('w:tblStyle');
            $xmlWriter->writeAttribute('w:val', $style);
            $xmlWriter->endElement();
            if (null !== $this->width) {
                $this->writeTblWidth($xmlWriter, 'w:tblW', TblWidth::PERCENT, $this->width);
            }
            $xmlWriter->endElement();
        }
    }

    /**
     * Write full style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeStyle(XMLWriter $xmlWriter, TableStyle $style)
    {
        // w:tblPr
        $xmlWriter->startElement('w:tblPr');

        // Table alignment
        if ('' !== $style->getAlignment()) {
            $tableAlignment = new TableAlignment($style->getAlignment());
            $xmlWriter->startElement($tableAlignment->getName());
            foreach ($tableAlignment->getAttributes() as $attributeName => $attributeValue) {
                $xmlWriter->writeAttribute($attributeName, $attributeValue);
            }
            $xmlWriter->endElement();
        }

        $this->writeTblWidth($xmlWriter, 'w:tblW', $style->getUnit(), $style->getWidth());
        $this->writeTblWidth($xmlWriter, 'w:tblCellSpacing', TblWidth::TWIP, $style->getCellSpacing());
        $this->writeIndent($xmlWriter, $style);
        $this->writeLayout($xmlWriter, $style->getLayout());

        // Position
        $styleWriter = new TablePosition($xmlWriter, $style->getPosition());
        $styleWriter->write();

        //Right to left
        $xmlWriter->writeElementIf($style->isBidiVisual() !== null, 'w:bidiVisual', 'w:val', $this->writeOnOf($style->isBidiVisual()));

        $this->writeMargin($xmlWriter, $style);
        $this->writeBorder($xmlWriter, $style);

        $xmlWriter->endElement(); // w:tblPr

        $this->writeShading($xmlWriter, $style);

        // First row style
        $firstRow = $style->getFirstRow();
        if ($firstRow instanceof TableStyle) {
            $this->writeFirstRow($xmlWriter, $firstRow);
        }
    }

    /**
     * Enable/Disable automatic resizing of the table
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $layout autofit / fixed
     */
    private function writeLayout(XMLWriter $xmlWriter, $layout)
    {
        $xmlWriter->startElement('w:tblLayout');
        $xmlWriter->writeAttribute('w:type', $layout);
        $xmlWriter->endElement(); // w:tblLayout
    }

    /**
     * Write margin.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeMargin(XMLWriter $xmlWriter, TableStyle $style)
    {
        if ($style->hasMargin()) {
            $xmlWriter->startElement('w:tblCellMar');

            $styleWriter = new MarginBorder($xmlWriter);
            $styleWriter->setSizes($style->getCellMargin());
            $styleWriter->write();

            $xmlWriter->endElement(); // w:tblCellMar
        }
    }

    /**
     * Write border.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeBorder(XMLWriter $xmlWriter, TableStyle $style)
    {
        if ($style->hasBorder()) {
            $xmlWriter->startElement('w:tblBorders');

            $styleWriter = new MarginBorder($xmlWriter);
            $styleWriter->setSizes($style->getBorderSize());
            $styleWriter->setColors($style->getBorderColor());
            $styleWriter->write();

            $xmlWriter->endElement(); // w:tblBorders
        }
    }

    /**
     * Writes a table width
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param string $elementName
     * @param string $unit
     * @param int|float $width
     */
    private function writeTblWidth(XMLWriter $xmlWriter, $elementName, $unit, $width = null)
    {
        if (null === $width) {
            return;
        }
        $xmlWriter->startElement($elementName);
        $xmlWriter->writeAttributeIf(null !== $width, 'w:w', $width);
        $xmlWriter->writeAttribute('w:type', $unit);
        $xmlWriter->endElement();
    }

    /**
     * Write row style.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeFirstRow(XMLWriter $xmlWriter, TableStyle $style)
    {
        $xmlWriter->startElement('w:tblStylePr');
        $xmlWriter->writeAttribute('w:type', 'firstRow');
        $xmlWriter->startElement('w:tcPr');

        $this->writeBorder($xmlWriter, $style);
        $this->writeShading($xmlWriter, $style);

        $xmlWriter->endElement(); // w:tcPr
        $xmlWriter->endElement(); // w:tblStylePr
    }

    /**
     * Write shading.
     *
     * @param \PhpOffice\PhpWord\Shared\XMLWriter $xmlWriter
     * @param \PhpOffice\PhpWord\Style\Table $style
     */
    private function writeShading(XMLWriter $xmlWriter, TableStyle $style)
    {
        if (null !== $style->getShading()) {
            $xmlWriter->startElement('w:tcPr');

            $styleWriter = new Shading($xmlWriter, $style->getShading());
            $styleWriter->write();

            $xmlWriter->endElement();
        }
    }

    /**
     * Set width.
     *
     * @param int $value
     */
    public function setWidth($value = null)
    {
        $this->width = $value;
    }

    /**
     * @param XMLWriter $xmlWriter
     * @param TableStyle $style
     */
    private function writeIndent(XMLWriter $xmlWriter, TableStyle $style)
    {
        $indent = $style->getIndent();

        if ($indent === null) {
            return;
        }

        $this->writeTblWidth($xmlWriter, 'w:tblInd', $indent->getType(), $indent->getValue());
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/TablePosition.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

/**
 * TablePosition style writer
 */
class TablePosition extends AbstractStyle
{
    /**
     * Write style.
     */
    public function write()
    {
        $style = $this->getStyle();
        if (!$style instanceof \PhpOffice\PhpWord\Style\TablePosition) {
            return;
        }

        $values = array();
        $properties = array(
          'leftFromText',
          'rightFromText',
          'topFromText',
          'bottomFromText',
          'vertAnchor',
          'horzAnchor',
          'tblpXSpec',
          'tblpX',
          'tblpYSpec',
          'tblpY',
        );
        foreach ($properties as $property) {
            $method = 'get' . $property;
            if (method_exists($style, $method)) {
                $values[$property] = $style->$method();
            }
        }
        $values = array_filter($values);

        if ($values) {
            $xmlWriter = $this->getXmlWriter();
            $xmlWriter->startElement('w:tblpPr');
            foreach ($values as $property => $value) {
                $xmlWriter->writeAttribute('w:' . $property, $value);
            }
            $xmlWriter->endElement();
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007/Style/TextBox.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer\Word2007\Style;

use PhpOffice\PhpWord\Style\TextBox as TextBoxStyle;

/**
 * TextBox style writer
 *
 * @since 0.11.0
 */
class TextBox extends Frame
{
    /**
     * Writer inner margin.
     */
    public function writeInnerMargin()
    {
        $style = $this->getStyle();
        if (!$style instanceof TextBoxStyle || !$style->hasInnerMargins()) {
            return;
        }

        $xmlWriter = $this->getXmlWriter();
        $margins = implode(', ', $style->getInnerMargin());

        $xmlWriter->writeAttribute('inset', $margins);
    }

    /**
     * Writer border.
     */
    public function writeBorder()
    {
        $style = $this->getStyle();
        if (!$style instanceof TextBoxStyle) {
            return;
        }
        $xmlWriter = $this->getXmlWriter();

        $xmlWriter->startElement('v:stroke');
        $xmlWriter->writeAttributeIf($style->getBorderSize() !== null, 'weight', $style->getBorderSize() . 'pt');
        $xmlWriter->writeAttributeIf($style->getBorderColor() !== null, 'color', $style->getBorderColor());
        $xmlWriter->endElement(); // v:stroke
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/Word2007.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

use PhpOffice\PhpWord\Element\Section;
use PhpOffice\PhpWord\Media;
use PhpOffice\PhpWord\PhpWord;
use PhpOffice\PhpWord\Shared\ZipArchive;

/**
 * Word2007 writer
 */
class Word2007 extends AbstractWriter implements WriterInterface
{
    /**
     * Content types values
     *
     * @var array
     */
    private $contentTypes = array('default' => array(), 'override' => array());

    /**
     * Document relationship
     *
     * @var array
     */
    private $relationships = array();

    /**
     * Create new Word2007 writer
     *
     * @param \PhpOffice\PhpWord\PhpWord
     */
    public function __construct(PhpWord $phpWord = null)
    {
        // Assign PhpWord
        $this->setPhpWord($phpWord);

        // Create parts
        $this->parts = array(
            'ContentTypes'   => '[Content_Types].xml',
            'Rels'           => '_rels/.rels',
            'DocPropsApp'    => 'docProps/app.xml',
            'DocPropsCore'   => 'docProps/core.xml',
            'DocPropsCustom' => 'docProps/custom.xml',
            'RelsDocument'   => 'word/_rels/document.xml.rels',
            'Document'       => 'word/document.xml',
            'Comments'       => 'word/comments.xml',
            'Styles'         => 'word/styles.xml',
            'Numbering'      => 'word/numbering.xml',
            'Settings'       => 'word/settings.xml',
            'WebSettings'    => 'word/webSettings.xml',
            'FontTable'      => 'word/fontTable.xml',
            'Theme'          => 'word/theme/theme1.xml',
            'RelsPart'       => '',
            'Header'         => '',
            'Footer'         => '',
            'Footnotes'      => '',
            'Endnotes'       => '',
            'Chart'          => '',
        );
        foreach (array_keys($this->parts) as $partName) {
            $partClass = get_class($this) . '\\Part\\' . $partName;
            if (class_exists($partClass)) {
                /** @var \PhpOffice\PhpWord\Writer\Word2007\Part\AbstractPart $part Type hint */
                $part = new $partClass();
                $part->setParentWriter($this);
                $this->writerParts[strtolower($partName)] = $part;
            }
        }

        // Set package paths
        $this->mediaPaths = array('image' => 'word/media/', 'object' => 'word/embeddings/');
    }

    /**
     * Save document by name.
     *
     * @param string $filename
     */
    public function save($filename = null)
    {
        $filename = $this->getTempFile($filename);
        $zip = $this->getZipArchive($filename);
        $phpWord = $this->getPhpWord();

        // Content types
        $this->contentTypes['default'] = array(
            'rels' => 'application/vnd.openxmlformats-package.relationships+xml',
            'xml'  => 'application/xml',
        );

        // Add section media files
        $sectionMedia = Media::getElements('section');
        if (!empty($sectionMedia)) {
            $this->addFilesToPackage($zip, $sectionMedia);
            $this->registerContentTypes($sectionMedia);
            foreach ($sectionMedia as $element) {
                $this->relationships[] = $element;
            }
        }

        // Add header/footer media files & relations
        $this->addHeaderFooterMedia($zip, 'header');
        $this->addHeaderFooterMedia($zip, 'footer');

        // Add header/footer contents
        $rId = Media::countElements('section') + 6; // @see Rels::writeDocRels for 6 first elements
        $sections = $phpWord->getSections();
        foreach ($sections as $section) {
            $this->addHeaderFooterContent($section, $zip, 'header', $rId);
            $this->addHeaderFooterContent($section, $zip, 'footer', $rId);
        }

        $this->addNotes($zip, $rId, 'footnote');
        $this->addNotes($zip, $rId, 'endnote');
        $this->addComments($zip, $rId);
        $this->addChart($zip, $rId);

        // Write parts
        foreach ($this->parts as $partName => $fileName) {
            if ($fileName != '') {
                $zip->addFromString($fileName, $this->getWriterPart($partName)->write());
            }
        }

        // Close zip archive and cleanup temp file
        $zip->close();
        $this->cleanupTempFile();
    }

    /**
     * Get content types
     *
     * @return array
     */
    public function getContentTypes()
    {
        return $this->contentTypes;
    }

    /**
     * Get content types
     *
     * @return array
     */
    public function getRelationships()
    {
        return $this->relationships;
    }

    /**
     * Add header/footer media files, e.g. footer1.xml.rels.
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param string $docPart
     */
    private function addHeaderFooterMedia(ZipArchive $zip, $docPart)
    {
        $elements = Media::getElements($docPart);
        if (!empty($elements)) {
            foreach ($elements as $file => $media) {
                if (count($media) > 0) {
                    if (!empty($media)) {
                        $this->addFilesToPackage($zip, $media);
                        $this->registerContentTypes($media);
                    }

                    /** @var \PhpOffice\PhpWord\Writer\Word2007\Part\AbstractPart $writerPart Type hint */
                    $writerPart = $this->getWriterPart('relspart')->setMedia($media);
                    $zip->addFromString("word/_rels/{$file}.xml.rels", $writerPart->write());
                }
            }
        }
    }

    /**
     * Add header/footer content.
     *
     * @param \PhpOffice\PhpWord\Element\Section &$section
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param string $elmType header|footer
     * @param int &$rId
     */
    private function addHeaderFooterContent(Section &$section, ZipArchive $zip, $elmType, &$rId)
    {
        $getFunction = $elmType == 'header' ? 'getHeaders' : 'getFooters';
        $elmCount = ($section->getSectionId() - 1) * 3;
        $elements = $section->$getFunction();
        /** @var \PhpOffice\PhpWord\Element\AbstractElement $element Type hint */
        foreach ($elements as &$element) {
            $elmCount++;
            $element->setRelationId(++$rId);
            $elmFile = "{$elmType}{$elmCount}.xml"; // e.g. footer1.xml
            $this->contentTypes['override']["/word/$elmFile"] = $elmType;
            $this->relationships[] = array('target' => $elmFile, 'type' => $elmType, 'rID' => $rId);

            /** @var \PhpOffice\PhpWord\Writer\Word2007\Part\AbstractPart $writerPart Type hint */
            $writerPart = $this->getWriterPart($elmType)->setElement($element);
            $zip->addFromString("word/$elmFile", $writerPart->write());
        }
    }

    /**
     * Add footnotes/endnotes
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param int &$rId
     * @param string $noteType
     */
    private function addNotes(ZipArchive $zip, &$rId, $noteType = 'footnote')
    {
        $phpWord = $this->getPhpWord();
        $noteType = ($noteType == 'endnote') ? 'endnote' : 'footnote';
        $partName = "{$noteType}s";
        $method = 'get' . $partName;
        $collection = $phpWord->$method();

        // Add footnotes media files, relations, and contents
        /** @var \PhpOffice\PhpWord\Collection\AbstractCollection $collection Type hint */
        if ($collection->countItems() > 0) {
            $media = Media::getElements($noteType);
            $this->addFilesToPackage($zip, $media);
            $this->registerContentTypes($media);
            $this->contentTypes['override']["/word/{$partName}.xml"] = $partName;
            $this->relationships[] = array('target' => "{$partName}.xml", 'type' => $partName, 'rID' => ++$rId);

            // Write relationships file, e.g. word/_rels/footnotes.xml
            if (!empty($media)) {
                /** @var \PhpOffice\PhpWord\Writer\Word2007\Part\AbstractPart $writerPart Type hint */
                $writerPart = $this->getWriterPart('relspart')->setMedia($media);
                $zip->addFromString("word/_rels/{$partName}.xml.rels", $writerPart->write());
            }

            // Write content file, e.g. word/footnotes.xml
            $writerPart = $this->getWriterPart($partName)->setElements($collection->getItems());
            $zip->addFromString("word/{$partName}.xml", $writerPart->write());
        }
    }

    /**
     * Add comments
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param int &$rId
     */
    private function addComments(ZipArchive $zip, &$rId)
    {
        $phpWord = $this->getPhpWord();
        $collection = $phpWord->getComments();
        $partName = 'comments';

        // Add comment relations and contents
        /** @var \PhpOffice\PhpWord\Collection\AbstractCollection $collection Type hint */
        if ($collection->countItems() > 0) {
            $this->relationships[] = array('target' => "{$partName}.xml", 'type' => $partName, 'rID' => ++$rId);

            // Write content file, e.g. word/comments.xml
            $writerPart = $this->getWriterPart($partName)->setElements($collection->getItems());
            $zip->addFromString("word/{$partName}.xml", $writerPart->write());
        }
    }

    /**
     * Add chart.
     *
     * @param \PhpOffice\PhpWord\Shared\ZipArchive $zip
     * @param int &$rId
     */
    private function addChart(ZipArchive $zip, &$rId)
    {
        $phpWord = $this->getPhpWord();

        $collection = $phpWord->getCharts();
        $index = 0;
        if ($collection->countItems() > 0) {
            /** @var \PhpOffice\PhpWord\Element\Chart $chart */
            foreach ($collection->getItems() as $chart) {
                $index++;
                $rId++;
                $filename = "charts/chart{$index}.xml";

                // ContentTypes.xml
                $this->contentTypes['override']["/word/{$filename}"] = 'chart';

                // word/_rels/document.xml.rel
                $this->relationships[] = array('target' => $filename, 'type' => 'chart', 'rID' => $rId);

                // word/charts/chartN.xml
                $chart->setRelationId($rId);
                $writerPart = $this->getWriterPart('Chart');
                $writerPart->setElement($chart);
                $zip->addFromString("word/{$filename}", $writerPart->write());
            }
        }
    }

    /**
     * Register content types for each media.
     *
     * @param array $media
     */
    private function registerContentTypes($media)
    {
        foreach ($media as $medium) {
            $mediumType = $medium['type'];
            if ($mediumType == 'image') {
                $extension = $medium['imageExtension'];
                if (!isset($this->contentTypes['default'][$extension])) {
                    $this->contentTypes['default'][$extension] = $medium['imageType'];
                }
            } elseif ($mediumType == 'object') {
                if (!isset($this->contentTypes['default']['bin'])) {
                    $this->contentTypes['default']['bin'] = 'application/vnd.openxmlformats-officedocument.oleObject';
                }
            }
        }
    }
}

## File: word/PHP_Word/vendor/phpoffice/phpword/src/PhpWord/Writer/WriterInterface.php

<?php
/**
 * This file is part of PHPWord - A pure PHP library for reading and writing
 * word processing documents.
 *
 * PHPWord is free software distributed under the terms of the GNU Lesser
 * General Public License version 3 as published by the Free Software Foundation.
 *
 * For the full copyright and license information, please read the LICENSE
 * file that was distributed with this source code. For the full list of
 * contributors, visit https://github.com/PHPOffice/PHPWord/contributors.
 *
 * @see         https://github.com/PHPOffice/PHPWord
 * @copyright   2010-2018 PHPWord contributors
 * @license     http://www.gnu.org/licenses/lgpl.txt LGPL version 3
 */

namespace PhpOffice\PhpWord\Writer;

/**
 * Writer interface
 */
interface WriterInterface
{
    /**
     * Save PhpWord to file
     *
     * @param string $filename
     */
    public function save($filename = null);
}

## File: word/prepara_word.php

<?php
$test = new mysqli('localhost', 'usuario_admin','password', 'test', 3306);
$test->set_charset("utf8");
function EnLetras($n)
{
    $f = new \NumberFormatter("es-ES", NumberFormatter::SPELLOUT);
    $entero = implode('.', array_slice(explode('.', $n), 0, 1));
    $decimal = implode('.', array_slice(explode('.', $n), 1, 1));
    if ($decimal != 00) {
        $entero = implode('.', array_slice(explode('.', $n), 0, 1));
        $decimal = implode('.', array_slice(explode('.', $n), 1, 1));
        return strtoupper($f->format($entero) . " con " . $f->format($decimal));
    } else {
        $entero = implode('.', array_slice(explode('.', $n), 0, 1));
        return strtoupper($f->format($entero));
    }
}
function fecha_larga($fecha)
{
    $fecha = substr($fecha, 0, 10);
    $numeroDia = date('d', strtotime($fecha));
    $dia = date('l', strtotime($fecha));
    $mes = date('F', strtotime($fecha));
    $anio = date('Y', strtotime($fecha));
    $dias_ES = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");
    $dias_EN = array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday");
    $nombredia = str_replace($dias_EN, $dias_ES, $dia);
    $meses_ES = array("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre");
    $meses_EN = array("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December");
    $nombreMes = str_replace($meses_EN, $meses_ES, $mes);
    return $nombredia . " " . $numeroDia . " de " . $nombreMes . " del " . $anio;
}
$json = json_decode(file_get_contents('php://input'), true);

if (!empty($json['contrato'])) {
    $num_contrato = $json['contrato'];
    $qA = $test->query("SELECT * from contrato where num_contrato = '" . $num_contrato . "'");
    $rA = $qA->fetch_row();
    $plantilla = 'CONTRATO PAGARE PLANTILLA.docx';
    $num_pagare = $rA[8];
    $fecha_pagare = fecha_larga($rA[10]);
    $adeuda_usd = EnLetras($rA[9]) . ' ( ' . $rA[9] . ' ) DOLARES AMERICANOS';
    $importe_usd = EnLetras($rA[5]) . ' ( ' . $rA[5] . ' ) DOLARES AMERICANOS';
    $cuota_adm_usd = EnLetras($rA[6]) . ' ( ' . $rA[6] . ' ) DOLARES AMERICANOS';
    $nombre_afiliado = $rA[1];
    $dni_afiliado = $rA[2];
    $dirección_afiliado = $rA[3];
    $a=$rA[4] == 1 ? ' AÑO ' : ' AÑOS ';
    $anos_afiliacion = EnLetras($rA[4]) . ' ( ' . $rA[4] . ' ) ' . $a;
    $fecha_contrato = fecha_larga($rA[7]);

    require "PHP_Word/vendor/autoload.php";
    $templateProcessor = new \PhpOffice\PhpWord\TemplateProcessor($plantilla);
    $templateProcessor->setValue(
        array(
            'num_pagare', 'fecha_pagare', 'adeuda_usd', 'importe_usd', 'cuota_adm_usd',
            'num_contrato', 'nombre_afiliado', 'dni_afiliado', 'dirección_afiliado',
            'anos_afiliacion', 'fecha_contrato'
        ),
        array(
            $num_pagare, $fecha_pagare, $adeuda_usd, $importe_usd, $cuota_adm_usd,
            $num_contrato, $nombre_afiliado, $dni_afiliado, $dirección_afiliado,
            $anos_afiliacion, $fecha_contrato
        )
    );
    $salida = 'contrato_' . $num_contrato . '.docx';
    $templateProcessor->saveAs($salida);
    echo $salida;
}