OpenXML Packaging API Test Design Spec

**Microsoft Confidential Office.Net Test Design Spec**

|  |  |
| --- | --- |
| Document last updated: | 10/17/2003 |
| Document created: | 6/13/2007 |
| Tester | WendyWu; HuiZeng; Changx |
| Program Manager | LaBrowne |
| Developer | DonghZ; ShDong; HaiyangG |
| Operations Contact |  |
| Other test contacts |  |
| Functional Specification(s) |  |
| PS Database/Area/Subarea | Office 14\Current\East Asia Office\Open XML API\ |
| Websites; Discussion Groups |  |
| Previous Owners (Test, PM, Dev) |  |

# Table of Contents

[OpenXML Packaging API Test Design Spec 1](#_Toc164121526)

[Table of Contents 1](#_Toc164121527)

[Feature Overview and Scope 2](#_Toc164121528)

[Feature Implementation Details 2](#_Toc164121529)

[Testing Strategy Overview 3](#_Toc164121530)

[Overview of your test approach 3](#_Toc164121531)

[Automation Strategy 3](#_Toc164121532)

[Dependencies 3](#_Toc164121533)

[Risks and contingency plans 3](#_Toc164121534)

[Test Area Breakdown 3](#_Toc164121535)

[Low-lever API 3](#_Toc164121536)

[Package Class 3](#_Toc164121537)

[Part Class 4](#_Toc164121538)

[Partial Class 4](#_Toc164121539)

[Constraints 5](#_Toc164121540)

[High-Level API 5](#_Toc164121541)

[User Scenarios 5](#_Toc164121542)

[Setup 6](#_Toc164121543)

[Specialty Test Issues 6](#_Toc164121544)

[Programmability: VBA / Macro / Scripting Command Interfaces 6](#_Toc164121545)

[Security/Privacy 6](#_Toc164121546)

[International Sufficiency 6](#_Toc164121547)

[Localization 6](#_Toc164121548)

[Accessibility 6](#_Toc164121549)

[Configuration / Printing 7](#_Toc164121550)

[Setup 7](#_Toc164121551)

[Service deployment and Operations 7](#_Toc164121552)

[Performance, Scalability, and Reliability 8](#_Toc164121553)

[User Assistance and documentation 9](#_Toc164121554)

[Other Test Collateral 9](#_Toc164121555)

[TDS History 9](#_Toc164121556)

# Feature Overview and Scope

**Office Open XML Format,** the new Office file format is built on a standard ZIP-based packaging model. In this format the content of each Office file type is broken down into smaller logical *parts* that are packaged together into a *ZIP archive.* Each individual part represents a particular aspect of the file that a user created by using Office application.

OpenXML Packaging API is a strong typed API which used to manipulate each part in Office Open XML format file. With this API, developers could be able to scan, manipulate, and modify the internal structure of Office Open XML format files.

OpenXML Packaging API is just manipulate with parts in new Office Open XML format files, it will not change any content in each parts (xml files).

The goal of testing defined in this TDS is to ensure that the APIs could manipulate all OpenXML defined parts. We will ensure that the APIs can consume valid Office documents and that Office application can open documents created/modified by API.

# Feature Implementation Details

OpenXML Packaging API has 20 Framework classes, 3 Package classes, 80 Part classes. Most of class are thru code genarator

Ideas of things to discuss in this area include:

        Is this feature client-side code, server-side code, or both?

        Is server-side code run by Microsoft, or can users set up their own standalone servers?

        What end-user/server file(s) does the code reside in?

        Describe any specific algorithms, data structures, memory management issues, or background operations that are important to testing the feature but may not be obvious from the feature interface.

        Describe any dependencies on other features or system components.

        If your feature is a service, what are the XML interfaces to the service and how are the used?

        Are there any debug tools/hooks (e.g. simulating specific memory conditions or the entry of specific values) that would help test the feature?

        What files, registry settings, database configurations or schemas are necessary to test the feature?

        If the feature is a legacy area, briefly discuss code history, and specify what features are being updated for this version #.

# Testing Strategy Overview

This section should discuss *how* you will test the area-- what techniques, tools, and strategies you will use.

## Overview of test approach

* BVT

Implement basic scenarios to make sure API has no block issue and hook up API structure

* API structure review

Review each class and methods/properties to find the following issues:

Function lacking

Dup function

Confuse function

Expose more/less method

* Feature Level testing

Target on Part/Package class level, verify the functionality of exposed methods and properties.

There are 7 packages class and 81 parts classes, for detail category, please refer Test Breakdown section.

* Scenario based testing

Executing user task with API provided function based on scenarios provided by PM spec.

## Automation Strategy

BVT and Feature level testing will be automated.

For BVT cases, API method/properties will be called directly as like as real user does.

For feature level testing, test framework will be designed to perform test cases.

Here is test framework architecture:



These tasks need to do for the framework:

1. Analysis and prepare test files
2. Implement TestLib and test execution class
3. Determine Log and verification items
4. Design test cases

Optional automation item

Check API comprehensive by compare the classes with package.xsd

Performance bench mark for each method

## Dependencies

* System.IO.Packaging namespace in WinFX(.Net Framework 3.0)
* OpenXML packaging schema

## Risks and contingency plans

* API design change

At first, we need review each build design when build available.

Since API design is not stabilized, we need keep agile on testing strategy to fit the new design.

For automation, test framework should be flexible to contain API change issue

* Schedule

4/30 Test framework done

5/15 Test done

5/25 Setup Test/RTM

# Test Area Breakdown

## Low-lever API

### Package Class (7)

WordprocessingDocument

WordprocessingTemplateDocument

SpreadsheetDocument

SpreadsheetTemplateDocument

PresentationDocument

PresentationTemplateDocument

PresentationSlideshowDocument

Since API design is not solid, here is just a sample for a package class:

|  |  |
| --- | --- |
|  | **Scenario/Cases** |
| **Properties** |  |
| Childparts | 1. Manipulate all child parts |
| **Methods** |  |
| Open() | 1. Could open specific well-formatted OpenXML file, cover all overloads 2. Friendly return value for incorrect file |
| Flush() | 1. Could save modifications to opened file 2. Check Constraints\* 3. Negative: call before Open() called |
| Close() | 1. Saves and closes the package plus all underlying part streams. 2. Negative: call before Open() called |
| GetParts() | 1. Returns a collection of all the parts in the package. |
| GetPart() | 1. Returns the specific part. Should cover all 80+ parts. 2. For the part occurrence>1, will return a collection. 3. Negative: specify a part which package does not contain |
| CreatePart() | 1. Creates a new package part. Should cover all 80+ parts. Cover all overloads. 2. Check constraints\* |
| DeletePart() | 1. Delete specific part from the package 2. Negative: Delete part which not in the package |
| GetRelationships() | ? TBD |
| Other methods in API Framework | TBD |

### Part Class (81)

Categorize part by content:

**Document Information parts(6)**

File Properties(3)

CoreFilePropertiesPart

ExtendedFilePropertiesPart

CustomFilePropertiesPart

Thumbnail(1)

ThumbnailPart

DigitalSignature（2）

DigitalSignatureOriginPart

XmlSignaturePart

**Word(14)**

MainDocumentPart

MainDocumentTemplatePart

AlternativeFormatImportPart

CommentsPart

DocumentSettingsPart

EndnotesPart[[1]](#endnote-2)

FontTablePart

FooterPart

FootnotesPart

GlossaryDocumentPart

HeaderPart

NumberingDefinitionsPart

StyleDefinitionsPart

WebSettingsPart

**Excel(25)**

WorkbookPart

WorkbookTemplatePart

CalculationChainPart

ChartsheetPart

WorksheetCommentsPart

ConnectionsPart

CustomPropertyPart

CustomXmlMappingsPart

DialogsheetPart

DrawingsPart

ExternalWorkbookPart

CellMetadataPart

PivotTablePart

PivotCacheDefinitionPart

PivotCacheRecordsPart

QueryTablePart

SharedStringTablePart

WorkbookRevisionHeaderPart

WorkbookRevisionLogPart

WorkbookUserDataPart

SingleCellTablePart

WorkbookStylesPart

TableDefinitionPart

VolatileDependenciesPart

WorksheetPart

**PPT(15)**

PresentationPart

PresentationTemplatePart

PresentationSlideshowPart

CommentAuthorsPart

SlideCommentsPart

HandoutMasterPart

NotesMasterPart

NotesSlidePart

PresentationPropertiesPart

SlidePart

SlideLayoutPart

SlideMasterPart

SlideSyncDataPart

UserDefinedTagsPart

ViewPropertiesPart

**DrawingML(9)**

ChartPart

ChartDrawingPart

DiagramColorsPart

DiagramDataPart

DiagramLayoutPart

DiagramStylePart

ThemePart

ThemeOverridePart

TableStylesPart

**Shared(12)**

CustomXmlPart

CustomXmlPropertiesPart

EmbeddedControlPart

EmbeddedObjectPart

EmbeddedPackagePart

FontPart

SpreadsheetPrinterSettingsPart

WordprocessingPrinterSettingsPart

AudioPart

VideoPart

ImagePart

VmlDrawingPart

**Categorize parts by relationship structure:**

Top-Level Parts (3)

Which can only be contained in package, such as DoucmentPart, PresentationPart, SpreadsheetPart

Bottom-Level Parts (12)

Which could not contain child part. All Binary parts and few XML parts are Bottom-Level

Mid-Level Parts (66)

Parts which could contain child part and also be child of other parts

Since API design is not solid, here is just a sample for a package class:

|  |  |
| --- | --- |
|  | **Scenario/Cases** |
| **Properties** |  |
| ChildParts | 1. Could manipulate all child parts |
| **Methods** |  |
| AddParts() | 1. Could new a specific package, cover all overloads |
| RemovePart() | 1. Could open specific well-formatted OpenXML file, cover all overloads 2. Friendly return value for incorrect file |
| GetStream() | 1. Get the XML or Binary bits of the part. 2. Cover all overloads |
| GetRelationships() | ? TBD |
| Other methods in API Framework | TBD |

### Constraints

Verified API provide constrains follow schema definition.

#### Occurrence

Eg. Documents.xml can only occur once in WordPackage.

…

#### Sequence

Parts should be ordered in the right order as schema defined.

Eg: mainDocumentPart schema

<!--

Word Main Document Part

-->

<xsd:complexType name="mainDocumentPart" ofapi:apiBase="XmlPart" ofapi:apiType="MainDocumentPart">

<xsd:attribute name="sourceRelationship" type="xsd:string ofapi:apiName="SourceRelationship" fixed="" />

<xsd:sequence>

<xsd:element name="alternativeFormatImportPart" type="CT\_AlternativeFormatImportPart"

minOccurs="0" maxOccurs="unbounded" ofapi:apiName="AlternativeFormatImportParts" />

<xsd:element name="commentsPart" type="CT\_CommentsPart" />

<xsd:element name="documentSettingsPart" type="CT\_DocumentSettingsPart" />

<xsd:element name="endnotesPart" type="CT\_EndnotesPart" />

<xsd:element name="fontTablePart" type="CT\_FontTablePart" />

<xsd:element name="footerPart" type="CT\_FooterPart" />

<xsd:element name="footnotesPart" type="CT\_FootnotesPart" />

<xsd:element name="glossaryDocumentPart" type="CT\_GlossaryDocumentPart" />

<xsd:element name="headerPart" type="CT\_HeaderPart" />

<xsd:element name="numberingDefinitionsPart" type="CT\_NumberingDefinitionsPart" />

<xsd:element name="styleDefinitionsPart" type="CT\_StyleDefinitionsPart" />

<xsd:element name="webSettingsPart" type="CT\_WebSettingsPart" />

<xsd:element name="Part" type="CT\_Part" />

<xsd:element name="Part" type="CT\_Part" />

</xsd:sequence>

</xsd:complexType>

### Partial Class

TBD. This section will be designed when generated class is ready.

### 

## High-Level API

### User Scenarios

* + - Document Inspection and Redaction
      * + Remove VBA projects and convert file type before emailing
        + Remove comments, revisions before publishing
        + Insert headers/footers and watermarks
    - Document Assembly
      * + Create pitchbooks from slide libraries
        + Pull data out of SQL or LOBi servers and publish in OpenXML with Building Blocks
    - Document Validation
      * + Validate structure of parts
        + Validate XML
    - Document Lifecycle and Workflow
      * + Rules application
        + Transcoding of documents
    - Document Merge

## Setup

Setup Process

Install/Uninstall/Repair

Dependency check

Setup UI

EULA, Language…

Binary location

Registry settings

# Specialty Test Issues

The following areas are common testing concerns across most all features. Detail test issues with each area for your test area, using the suggestions with each area. You may wish to discuss your feature with your team’s contact for the below area to get further ideas. If the area is not applicable to your feature note “No issues with this test area.” in the section.

## Programmability: VBA / Macro / Scripting Command Interfaces

TBD. It should follow Office product standard. Such as passing FxCop check.

## Security/Privacy

Could the API run under a small privilege?

## International Sufficiency

ANSI/Unicode/DBCS URI/PartName as parameter

## Localization

No Localization issue for this API.

For Setup, TBD

## Accessibility

The API has no UI, so it has no accessibility issues.

## Configuration / Printing[[2]](#endnote-3)

No issues with this test area

## Setup

Dependency: WinFX

Integration

API could be integrated with IDE (VS2005)

## Service deployment and Operations

If your feature has server-side code you will want to discuss the feature with your team’s operations contact and/or your team’s system engineer. They can help you understand which of the following operations issues relate to your feature, and how best to test them:

#### Service Deployment

        List the files (i.e. code, data, settings) that are deployed and the servers they are deployed to.

        What database configurations/settings and/or server extensions are needed to deploy your service?

        Are deployment scripts used? If so, who owns testing them, and how will they be tested?

        How are you going to verify correct deployment of the service?

        If there is a failure during deployment, are the servers rolled back to their original state?

#### General Operations

        How will you test the replication and load-balancing of the feature?

        Does the feature have admin tools and/or how do operations personnel get access to the data, files, and settings?

         What is the latency of network response (performance) when you connect from different locations (US/Europe/Asia)?

        What is the latency of network response (performance) when you use different bandwidth connections (LAN, DSL, Cable, Dial-up)?

        How will you test that URLs, accounts, IP addresses, and machine names are not hard coded and that they reside in configurable resource files?

        How will you test that your feature is using a DSN (Data Source Name) when interacting with databases?

        What dependencies does your feature have on external-to-Office services?

#### Failure conditions

It’s best that the service doesn’t fail, however if it does, it’s critical that it fail in a graceful way that allows operations to quickly understand the problem, and recover with minimal interruption to users.

        How will you simulate back-end failures (SQL server, file server)?, front-end failures (web, mail)? Network connection failures? Third-party-dependency failures?

        Are there test scripts or monitoring information that are launched in response to errors?

        How will you test the recovery of the feature when errors occur?

        How will you test the rollback of actions when failures occur during a transaction? (e.g. no leaving partial data from a failed transaction in a db)

        How will you test that services are redirected when they’re down?

        How does client behavior change when offline and/or when the service is not available?

#### Monitoring and Instrumentation

        What data should be collected about usage of your feature? How will you test that your feature is logging the correct information?

        What monitoring information (e.g. PerfMon counters) is made available to operations to indicate the health of a server? How will you test that this data is working and accurate?

        Can logging or monitoring information be audited (having a way to test that logging and monitoring information is accurate)? If so, how will you test this?

        What ways can customers give feedback? How will you test these?

## Performance, Scalability, and Reliability

Discuss your feature with your team’s performance contact. They can help you understand which of the following performance issues relate to your feature, and how best to test them:

#### Client performance

        What modules, controls, features or content are loaded? Which of these items is most likely to affect performance?

        What is the baseline for performance? Is it measured against a similar control, feature, application, or service?

        Are there any test cases that should be part of the overall QnS performance scripts?

        Is there a possibility for performance slow downs due to new services or functionality?

        Which types of low resources testing (memory, hard drive, network bandwidth, connections, CPU, etc.) will you test with your area? How will you test or simulate the conditions?

        What are some real-world scenarios concerning realistic volumes of data, applications running, or durations that need to be tested to ensure a good client experience? How will you test or simulate these volumes?

#### Server performance

**        What are the top tasks that users of your feature will do?**

**        Consider how often your service will be used on a daily basis.**

**        Work with your PM and your test team’s performance contact to identify the following performance goals for your area:**

       **Throughput –** successful completions of user scenarios. For example, successful basket creations, orders processed, and searches performed are considered successful completions for an eCommerce site.  How many transactions per second does the server need to handle? How will you test this? Will the service tolerate lost requests or errors?

       Response time – How long will users wait for a response? Check <http://aceteam/library/perfdocs/responsetime.asp> regarding response times advice. What is the allowable latency? How will you test this?

       Concurrent users – How many concurrent users will the service support? How will you test this?

        Scalability: Integration can have a big impact on scalability of a service.  How does your service interact with other services or features?  What type of tests will you run to make sure the feature scales with a varying number of users, servers, processors, memory and network bandwidth?

## User Assistance and documentation

        Plan to review help topics for your feature.

        What questions are users most likely to have? Can help topics be found by asking these questions?

        What types of tasks will people perform that use your feature? Can help be found by searching for these tasks in the Assistance Center?

# Other Test Collateral

This is where you can place contact information and resources that don’t fit in the header of this document. Examples include:

        More links to resources such as test tools, websites, test cases, automation suites.

        Explanation of Raid information (group, area, subarea), if necessary.

        Reminders of information necessary to repro bugs in this feature (e.g. if it’s more important than usual to specify what browser/OS you were on).

        Background reading if the feature needs to conform to standards like WinLogo, RFC’s, W3C initiatives, etc.

        Test contacts for East Asian, Complex Scripts, or Ireland teams.

        Test, Dev or PM owners of related features.

        Definitions of jargon, acronyms or buzzwords used when discussing the area.

# TDS History

|  |  |  |
| --- | --- | --- |
| date | changes made | author |
| 4/9/07 | Created | HuiZeng |
| 4/19/07 | Update test breakdown | HuiZeng |
| 4/23/07 | Update automation plan | HuiZeng |

1. 尾注 test1 [↑](#endnote-ref-2)
2. Endnote 2 [↑](#endnote-ref-3)