Oracle® Outside In Web View Export

Developer's Guide

Release 8.5.0

E50797-01

August 2014

ORACLE

Oracle Outside In Web View Export Developer’s Guide, Release 8.5.0

E50797-01

Primary Author: Mike Manier

Preface

Web View Export is part of Oracle's family of Original Equipment Manufacturer (OEM) technologies known as Oracle Outside In Technology, a powerful document extraction, conversion and viewing technology that can access the information in more than 600 file formats.

Audience

This document is intended for software developers who are responsible for integrating Oracle Outside In Technology into their applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, go to:

http://www.oracle.com/technetwork/indexes/documentation/index.html#middleware

and click on Outside In Technology.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.

Convention	Meaning
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
Forward slashes (/)	Forward slashes are used to separate the directory levels in a path to a UNIX server, directory, or file. Forward slashes are also used to separate parts of an Internet address. A forward slash will always be included at the end of a UNIX directory name and might or might not be included at the end of an Internet address.
Backward slashes (\)	Backward slashes are used to separate the levels in a path to a Windows server, directory, or file. A backward slash will always be included at the end of a Windows server, directory, or file path.
<install_dir>/	This notation refers to the location on your system of the main product installation directory.

Part I

Getting Started with Web View Export

This section provides an introduction to the Web View Export SDK.

Part I contains the following chapters:

1 Introduction

Outside In Web View Export is an SDK that enables an application to produce high quality, web-compatible renditions of documents created by standard business software. When asked to render a document, Outside In Web View Export produces HTML, CSS, and Javascript files which, when displayed in a browser, provide a high fidelity, interactive, and customizable user experience.

There may be references to other Oracle Outside In Technology SDKs within this manual. To obtain complete documentation for any other Oracle Outside In product, see:

http://www.oracle.com/technetwork/indexes/documentation/index.html#middleware

and click on Outside In Technology.

This chapter includes the following sections:

1.1 What's New in this Release

- The updated list of supported formats is linked from the page http://www.outsideinsdk.com/. Look for the data sheet with the latest supported formats.

1.2 Web View Export Overview

The following sections provide an overview of the Web View Export SDK:

1.2.1 Browser Compatibility

The output produced by the Web View Export SDK takes advantage of the current state-of-the-art browsers and the capabilities of HTML5, sometimes including

Javascript or HTML5-specific features such as a drawing canvas, web fonts, or SVG elements. Therefore, the use of an HTML5-capable browser is assumed and required.

1.2.2 Architecture

The architecture of Outside In Web View Export consists of two major components:

- An export module (EXH5) with a robust API similar to other Outside In export products, which produces high-quality renditions of input documents.
- A Javascript library (OIT.JS) used by the output HTML files, which provides interactive functions and provides a browser-side API of its own that allows web developers to customize and extend the capabilities of the web view of a document.

In a typical deployment, the export module would be used on a server to create the Web View Export output. The output files, along with the Javascript library, would then be made available by a web server to users who are accessing the output from their browsers.

Export Module

The export module is a loadable library with a rich API for controlling its behavior and certain aspects of the output. Versions of the API are available for C, Java, and .NET programs.

Detailed reference documentation for the API can be found at the following locations:

- [Link for C Reference Manual]
- [Link for Java Reference Manual]
- [Link for .NET Reference Manual]

Web View Export Javascript Library

The Web View Export browser-side API consists of a global object that provides a set of functions that allow a script writer to customize the navigation and interactive behavior of the Web View Export output. Outside In Web View Export allows custom scripts to be referenced from within the HTML5 renditions, which enables use of the browser-side API and integration into a web application.

1.2.3 Web View Export Output

For each exported document, Web View Export will produce at least one HTML file, one data file in either Javascript or JSON format (depending on which output structure option is selected), and one CSS file. Depending on configuration settings and the contents of the input document, Web View Export may also produce PNG image files, TTF font files, plain text files, and additional HTML and/or JSON files.

Customizing the Web View Export Output

See Appendix 10, "Web View Export Options" for the C API of the Web View Export Module.

The descriptions below are intended to add context to some of the new options that have been added specifically for the Web View Export product.

Partitioning of Content Between Output Files (Output Paging)

It is likely that an application would not want to render large documents as a single HTML output file. The Web View Export export module provides an “output paging”

option that allows a document be rendered to separate HTML files for each page, allowing for faster random access in a browser.

- C SDK: SCCOPT_OUTPUT_STRUCTURE
- Java API:
- .NET API:

Extending Output with Custom Scripts

The export module can insert links to one or more custom scripts into the Web View Export output. This enables a custom script to interact in the browser with the Web View Export Javascript API, and allows a script to customize the ways in which the user interacts with the document.

Because the order in which scripts are loaded is significant, Web View Export allows complete control over the ordering of all script links in the output HTML. One option is provided for inserting scripts above the link to the Outside In Javascript library (oit.js), and another option is provided for inserting scripts after oit.js. If multiple scripts are linked, they will appear in the output file in the order they were specified to the export module.

- SCCOPT_PRE_LIBRARY_SCRIPT
- SCCOPT_POST_LIBRARY_SCRIPT

1.3 Definition of Terms

The following terms are used in this documentation.

Term	Definition
Developer	Someone integrating this technology into another technology or application. Most likely this is you, the reader.
Source File	The file the developer wishes to export.
Output File	The output file being written.
Data Access Module	The core of Oracle Outside In Data Access, in the SCCDA library.
Data Access Submodule (also referred to as "Submodule")	This refers to any of the Oracle Outside In Data Access modules, including SCCEX (Export), but excluding SCCDA (Data Access).
Document Handle (also referred to as "hDoc")	A Document Handle is created when a file is opened using Data Access (see Chapter 6, "Data Access Common Functions"). Each Document Handle may have any number of Subhandles.
Subhandle (also referred to as "hItem")	Any of the handles created by a Submodule's Open function. Every Subhandle has a Document Handle associated with it. For example, the hExport returned by EXOpenExport is a Subhandle. The DASetOption and DAGetOption functions in the Data Access Module may be called with any Subhandle or Document Handle. The DARetrieveDocHandle function returns the Document Handle associated with any Subhandle.

1.4 Directory Structure

Each Oracle Outside In product has an sdk directory, under which there is a subdirectory for each platform on which the product ships. Under each of these directories are the following three subdirectories:

- **docs**: Contains both a PDF and HTML version of the product manual.
- **redist**: Contains only the files that the customer is allowed to redistribute. These include all the compiled modules, filter support files, .xsd and .dtd files, cmmap000.bin, and third-party libraries.
- **sdk**: Contains the materials necessary to develop software that uses the Web View Export export module, as well as demonstration code for instructional purposes.

In the root directory of the SDK deliverable, there are two files:

- **README**: Explains the contents of the sdk, and that makedemo must be run in order to use the sample applications.
- **makedemo** (either .bat or .sh – platform-based): This script will either copy (on Windows) or Symlink (on Unix) the contents of …/redist into …/sdk/demo, so that sample applications can then be run out of the demo directory.

1.4.1 Installing Multiple SDKs

If you load more than one OIT SDK, you must copy files from the secondary installations into the top-level OIT SDK directory as follows:

- **docs** – copy all subdirectories named "[product name]guide" into this directory.
- **redist** – copy all binaries into this directory.
- **sdk** – this directory has several subdirectories: common, demo, lib, resource, samplecode, samplefiles. In each case, copy all of the files from the secondary installation into the top-level OIT SDK subdirectory of the same name. If the top-level OIT SDK directory lacks any directories found in the directory being copied from, just copy those directories over.

2

Implementation Issues

This chapter covers some issues specific to using the Export products.

This chapter includes the following sections:

2.1 Running in 24x7 Environments

To ensure robust 24x7 performance in server applications embedding the different export products, it is strongly recommended that the technology be run in a process separate from the server's primary process.

The file filtering technology underlying the technology represents almost a quarter of a million lines of code. This code is expected to robustly deal with any stream of bytes, of any length (any file), in all cases. Oracle has dedicated, and continues to dedicate, significant effort into making this technology extremely robust. However, in real world situations, expect that some small number of malformed files may force the filters into unstable states. This generally results in either a memory exception (which can be trapped and recovered from gracefully), infinite loop or a wild pointer that causes the filter to write into memory that is part of the same process but does not belong to the filter. In the latter situation, this wild pointer condition cannot be trapped.

On the desktop this is not a significant problem since the number of files being dealt with is relatively small. In a 24x7 server environment, however, a wild pointer can be extremely disruptive to the server process and produce serious problems. The best solution for dealing with this problem is to run any application that reads complex file formats in a separate process. This solution protects the application from the susceptibility of filtering technology to the unknown quality of input files.

It must be stressed that files that lead to wild pointers or infinite loops occur very infrequently, usually as a result of a third-party conversion process or beta versions of applications. Oracle is committed to addressing these issues and to updating and expanding its testing tools and corpus of documents to proactively minimize this "garbage in-garbage out" problem.

2.2 Running in Multiple Threads or Processes

On certain platforms, export products may be run in a multithreaded or multiprocessing application. The thing to remember when doing so is that each thread must go through all the steps listed in Chapter 1, "Introduction."

2.3 Web View Export Issues

Three issues have been identified when using Web View Export:

- If multiple pages of garbage output occur when exporting images, it is possible that the default setting of the SCCOPT_FALLBACKFORMAT option (FI_ASCII-8) is forcing the technology to attempt to read files that it cannot identify as text. Setting the pertinent option to the value FI_NONE (FallbackFormat on the server version) prevents the software from exporting unidentified binary files as though they were text.
- The SCCOPT_FONTDIRECTORY option must be set for Linux. On Windows, the SCCOPT_FONTDIRECTORY option may be used to indicate a font directory, but if the option is not set the Windows' system font directory will be used.
- Only TrueType fonts are supported in Web View Export at this time.

3

Sample Applications

Each of the sample applications included in this SDK is designed to highlight a specific aspect of the technology's functionality. We ship built versions of these sample applications. The compiled executables should be in the root directory where the product is installed.

The following copyright applies to all sample applications shipped with this product:

This chapter includes the followinig sections:

3.1 Building the Samples on a Windows System

Microsoft Visual Studio project files are provided for building each of the sample applications. For 32-bit versions of Windows, versions of the project files are provided for Visual Studio 6 (.dsp files) and Visual Studio 2005 (.vcproj files).

Because .vcproj files may not pick up the right compiler on their own, you need to make sure that you are building with the Win64 configuration in Visual Studio 2005. For 64-bit versions of Windows, only the Visual Studio 2005 versions are available.

The project files for the sample applications can be found in the samplecode\win subdirectory of the Oracle Outside In SDK.

For specific information about building the sample applications on your UNIX OS, see Chapter 5, "UNIX Implementation Details."

3.2 An Overview of the Sample Applications

Here's a quick tour of the sample applications provided with this product. Not all of the sample applications are provided for both the Windows and UNIX platforms. See the heading of each application's subsection for clarification.

This section includes the following sample applications:

- Section 3.2.2, "wv_sample_exporter"
- Section 3.2.3, "demoserver_go"
- Section 3.2.4, "exsimple"

3.2.1 demoserver_node.js

The sample application demoserver_node.js is a minimalist web server designed for testing the features of Outside In Web View Export. Its main use is to automate the conversion of input files via the Web View Export Sample Application wv_sample_exporter, and serve the results to a web browser.

Architecture

demoserver is implemented as a javascript file that is executed by node.js. Node.js is available on multiple platforms, and demoserver should work on Linux as well as Windows.

Installation

Installation requires the following steps:

1. Install node.js
2. Get demoserver.js and demoserver.config from the sample application directory demoserver_nodejs
3. Edit demoserver.config to configure demoserver. This part may be optional if you run the demoserver from the same directory where it appears in the Web View Export SDK. By default, it is configured to look for the Web View Export executable files in the demo subdirectory of the SDK.

Once configured, you run the server with the command line:

```
node demoserver.js
```

Configuration

At startup, demoserver.js will load demoserver.config and use the options as specified. These options are only read at startup time, so if you wish to change an option, you must restart demoserver.

Here is a sample configuration:

```
{
  port: 80,

  // map of virtual directories
  dirmap: [
    { vpath: '/samplefiles', fspath: '../../samplefiles' },
    { vpath: '/out/',        fspath: './output' },
    { vpath: '/assets/',     fspath: '../../assets'},
    { vpath: '/scripts/',    fspath: '../scripts'},
    ],

  export_config:
  {
  // exporter program
    exporter: "../../demo/wv_sample_exporter.exe",

  // (optional) extra parameters for exporter program, as an array of strings
    params: ["-r", "/assets/", "-s", "/scripts/demoapp.js"],
```

```
  // Output virtual directory
    output: "/out/",
  },
}
```

Note that this file is a Javascript object declaration, and will be directly interpreted as such at run time. If you break the Javascript syntax of the file, the server will exit with an error message. (It's actually Javascript as opposed to JSON format. The syntax is mostly the same, but because Javascript allows comments and JSON does not, Javascript was used.)

Remember these rules:

- No semicolons
- Backslashes in strings must be escaped as \\
- Either single or double quotes are allowed as string delimiters, but must be paired like-with-like
- Don't change the { and } at the beginning and end of the file.

Configuration Parameters

- **port**: [Optional, defaults to 8888] This is a number that specifies the port on which demoserver will be listening. If you use port 80, you can just point your browser to http://localhost (or http://127.0.0.1). Any other value will need to be specified in your browser's address bar; e.g., listening on port 666 would require http://localhost:666
- **dirmap**: [Required] This defines the set of virtual directories for your web server. In Javascript syntax terms, this setting is an array of objects, each of which contains a field vdir, which defines a virtual directory, and fspath, which tells the server what real directory to use when files from this virtual directory are requested.

  You may add any number of virtual directories, and a virtual subdirectory may be defined whose physical directory is not a subdirectory of the virtual parent's physical directory.

  - **export_config**: [Required for exporting] This is a collection of values that must be specified in order for the demoserver to be able to export files to HTML5. It consists of the following values:
  - **output**: [Required for exporting] This field's value must be a virtual directory defined in the dirmap. When exporting files, demoserver will tell the export program to put its output in this directory.
  - **exporter**: [Required for exporting] This is the path to the executable that will be used to export files. In theory this can be any command line application that accepts the input file as its first parameter and the output file name as its second parameter, but demoserver does assume that this program will be wv_sample_exporter, and generates additional command line parameters for this specific program.
  - **params**: [Optional, no default value] An array of strings. If you would like your exporter application to have any further parameters on its command line you can specify them here.
  - **prefix**: [Optional, no default value] This is a string which will be used as the prefix for your output file names. For example, if export_config.prefix is set to

'_', then when exporting a file named file.doc, the primary output file that will be specified to the exporter is _file.doc.html

- **defaultMime**: [Optional, defaults to 'application/octet-stream'] A string value that will be used as the default MIME type provided by the server for files whose file extension doesn't match a commonly used Web format.

filebrowse.html

This is a page supplied with demoserver_node.js that you can use to navigate directories on your file system to conveniently export documents at will.

3.2.2 wv_sample_exporter

The sample application wv_sample_exporter uses the Web View Export SDK to produce HTML5 output files. It has two required parameters, the first being the input file name and the second being the name of the primary output file; as in:

```
wv_sample_exporter InputFile OutputFile
```

The following command line switches are used with this application to modify the way the HTML5 output is produced.

Switch	Parameters	Function	SDK Function
-p	'f', 'c', or 's', indicating structure that is flat, chunked (Ajax), or streaming (Ajax)	Chooses the structure of the output files generated during the export process.	maps to value of SCCOPT_OUTPUT_STRUCTURE option
-r	ResourcePath	ResourcePath is the base URL for .js, .css, and .png assets.	maps to option SCCOPT_URLPATH_RESOURCES
-o	OutputPath	OutputPath is the base URL for links to generated output files.	maps to option SCCOPT_URLPATH_OUTPUT
-s	ScriptUrl	ScriptUrl is a the URL of a javascript file to be linked from the output files.	maps to option SCCOPT_POST_LIBRARY_SCRIPT
-c	CssUrl	CssUrl is the URL of a CSS file to be linked from the output files.	maps to option SCCOPT_EXTERNAL_STYLESHEET
-n	ArchiveNode	ArchiveNode indentifies a node from the input file, to be used as the exported document (applies only to archive files, e.g., .zip)	maps to a record ID parameter passed into DAOpenTreeRecord
-a	AttachmentId	AttachmentId identifies an attachment from the input file, to be used as the exported document (applies only to email message files.)	maps to a subdoc id value to be used in a call to DAOpenSubdocumentById with the id type of DASUBDOCID_ATTACHMENTINDEX
-k	Key, Value	Key and Value are a key/value pair to be made available to scripts	maps to EXAddKeyValue
-f	FontDirectory	FontDirectory indicates where the exporter should find fonts to use.	maps to option SCCOPT_FONTDIRECTORY

3.2.3 demoserver_go

This is a sample server application very similar to the demoserver for node.js. It is written in the Go programming language; compiled executables are included as well as source code. Please see the readme.txt file in this application's directory for more information.

3.2.4 exsimple

This simple command line driven program allows the user to run a single source file through the software. The user can choose the source file, an output file and set the various options.

To run the program, type:

```
exsimple in_file out_file config_file
```

- *in_file* is the input file to be converted
- *out_file* is the output location
- *config_file* is the configuration file that sets the conversion options. If no configuration file is specified, default.cfg in the current directory is used.

The configuration file is a text file used to set the conversion options. We recommend reading through the configuration file for more information about valid options and their values (use of invalid options results in exsimple not producing output).

Follow these instructions to set configurable options.

- Set the Output ID to FI_HTML5 before running the software.
- It is also recommended that you set SCCOPT_FALLBACKFORMAT to FI_NONE. This prevents the export of unidentified binary files as though they were text, which could generate pages of garbage output.
- It is required that the "fontdirectory" section of the configuration file be set to point to a valid font directory.

Part II

Using the C/C++ API

This section provides details about using the SDK with the C/C++ API.

Part II contains the following chapters:

4

Windows Implementation Details

The Windows implementation of this software is delivered as a set of DLLs. For a list of the currently supported platforms, see:

http://www.oracle.com/technetwork/indexes/documentation/index.html#middleware

Click on Outside In Technology. On the Getting Started tab of the Outside In library, click on the current release number under the Certified Platforms heading.

This chapter includes the following sections:

4.1 Installation

To install the demo version of the SDK, copy the contents of the ZIP archive (available on the web site) to a local directory of your choice.

This product requires the Visual C++ libraries included in the Visual C++ Redistributable Package available from Microsoft. There is a version of this package for the x86 version of Windows. This can be downloaded from www.microsoft.com, by searching on the site for the following package:

- vcredist_x86.exe

The required download version is the "2005 SP1 Redistributable Package."

Oracle Outside In requires the msvcr80.dll redistributable module.

The installation directory should contain the following directory structure.

Directory	Description
\redist	Contains a working copy of the technology.
\sdk\assets	Contains Javascript, CSS, and image files that are used by the Web View Export output.
\sdk\common	Contains the C include files needed to build or rebuild the technology.

Directory	Description
\sdk\demo	Contains the compiled executables of the sample applications.
\sdk\lib	Contains the library (.lib) files needed for the products.
\sdk\resource	Contains localization resource files.
\sdk\samplecode	Contains a subdirectory holding the source code for a sample application.
\sdk\samplefiles	Contains sample files designed to exercise the technology.

4.1.1 NSF Support

Notes Storage Format (NSF) files are produced by the Lotus Notes Client or the Lotus Domino server. The NSF filter is the only Oracle Outside In filter that requires the native application to be present to filter the input documents. Due to integration with an outside application, NSF support will not work with redirected I/O, when an NSF file is embedded in another file, or with IOTYPE_UNICODEPATH. Either Lotus Notes version 8 or Lotus Domino version 8 must be installed on the same machine as OIT. A 32-bit version of the Lotus software must be used if you are using a 32-bit version of OIT. A 64-bit version of the Lotus software must be used if you are using a 64-bit version of OIT. On Windows, SCCOPT_LOTUSNOTESDIRECTORY should be set to the directory containing the nnotes.dll. NSF support is only available on the Win32, Win x86-64, Linux x86-32, and Solaris Sparc 32 platforms.

4.2 Libraries and Structure

The following is an overview of the files in the main installation directory for all five Oracle Outside In export products.

4.2.1 API DLLs

These libraries implement the API. They should be linked with the developer's application. Files with a .lib extension are included in the SDK.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
sccda.dll	Data Access module	X	X	X	X	X	X
sccex.dll	Export module	X	X	X	X	X	X
sccfi.dll	File Identification module (identifies files based on their contents).	X	X	X	X	X	X

The File ID Specification may not be used directly by any application or workflow without it being separately licensed expressly for that purpose.

4.2.2 Support DLLs

The following libraries are used for support.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
ccflex.dll	A data model adapter that converts from stream model utilized by Oracle Outside In filters to the FlexionDoc Tree model used as a basis by XML Export.					X	
libexpatw.dll	A third-part XML parser					X	
ocemul.dll	Output component emulation module	X	X	X	X	X	X
oswin*.dll	Interface to the native GDI implementation oswin32.dll is the 32-bit version, oswin64.dll is the 64-bit version	X	X	X	X	X	
sccanno.dll	The annotation module	X	X				X
sccca.dll	Content Access module (provides organized chunker data for the developer)	X	X	X			X
sccch.dll	Chunker (provides caching of and access to filter data for the export engines)	X	X	X	X	X	X
sccdu.dll	Display Utilities module (includes text formatting)	X	X	X	X	X	X
sccexind.dll	The core engine for all Search Export formats: SearchText, SearchHTML, SearchML and PageML			X	X		
sccfmt.dll	Formatting module (resolves numbers to formatted strings)	X	X		X	X	X
sccfut.dll	Filter utility module	X	X	X	X	X	X
sccind.dll	Indexing engine. In Search Export, it handles common functionality.	X	X	X	X		X
scclo.dll	Localization library (all strings, menus, dialogs and dialog procedures reside here)	X	X	X	X	X	X
sccole2.dll	OLE rendering module	X	X	X	X	X	X
sccsd.dll	Schema Definition Module Manager (brokers multiple Schema Definition Modules)			X		X	
sccut.dll	Utility functions, including IO subsystem	X	X	X	X	X	X
sccxt.dll	XTree module					X	
sdflex.dll	Schema Definition module (handles conversion of XML string names and attribute values to compact binary representations and vice versa)			X		X	
wvcore.dll	The GDI Abstraction layer	X	X		X	X	X

4.2.3 Engine Libraries

The following libraries are used for display purposes.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
debmp.dll	Raster rendering engine (TIFF, GIF, BMP, PNG, PCX…)			X		X	
devect.dll	Vector/Presentation rendering engine (PowerPoint, Impress, Freelance…)	X	X	X		X	X
dess.dll	Spreadsheet/Database (Excel, Calc, Lotus 123…)		X	X		X	
detree.dll	Archive (ZIP, GZIP, TAR…)		X	X			
dewp.dll	Document (Word, Writer, WordPerfect…)		X	X	X		X

4.2.4 Filter and Export Filter Libraries

The following libraries are used for filtering.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
vs*.dll	Filters for specific file types (there are more than 150 of these filters, covering more than 600 file formats)	X	X	X	X	X	X
oitnsf.id	Support file for the vsnsf filter.	X	X	X	X	X	X
exgdsf.dll	Export filter for GIF, JPEG, and PNG graphics files	X				X	X
eximg.dll	Extended image conversion module		X				
exh5.dll	Export filter for HTML5 files						X
exhtml.dll	Export filter for HTML files	X					
exihtml.dll	Export filter for SearchHTML				X		
exitext.dll	Export filter for SearchText				X		
exixml.dll	Export filters for XML files using the SearchML schema				X		
expage.dll	Export filter for XML files using the PageML schema				X		
expagelayout.dll	Page layout module			X			
exxml.dll	XML Export module					X	
sccimg.dll	Image conversion module	X	X			X	X

4.2.5 Premier Graphics Filters

The following are graphics filters.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
i*2.flt	30 .flt files (import filters for premier graphics formats)	X	X	X	X	X	X

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
isgdi32.dll	Interface to premier graphics filters	X	X	X	X	X	X

4.2.6 Additional Files

The following files are also used.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
adinit.dat	Support file for the **vsacad** filter	X	X	X	X	X	X
cmmap000.bin	Tables for character mapping (all character sets)	X	X	X	X	X	X
cmmap000.sbc	Tables for character mapping (single-byte character sets). This file is located in the /sdk/common directory.	X	X	X	X	X	X
cmmap000.dbc	Identical to **cmmap000.bin**, but renamed for clarity (.dbc = double-byte character). This file is located in the /sdk/common directory.	X	X	X	X	X	X
exbf.dll	Internal				X		
pageml.dtd	The Document Type Definition for the PageML schema				X		
pageml.xsd	The Extensible Schema Definition for the PageML schema				X		
searchml3.dtd	The Document Type Definitions for the SearchML schema				X		
searchml3.xsd	The Extensible Schema Definitions for the SearchML schema				X		
flexiondoc.dtd	The DTD version of the Flexiondoc schema					X	
flexiondoc.xsd	The schema version of the Flexiondoc schema					X	

4.3 The Basics

The following is a discussion of some basic usage and installation features.

All the steps outlined in this section are used in the sample applications provided with the SDK. Looking at the code for the **exsimple** sample application is recommended for those wishing to see a real-world example of this process.

4.3.1 What You Need in Your Source Code

Any source code that uses this product should `#include` the file sccex.h and `#define WINDOWS` and `WIN32` or `WIN64`. For example, a Windows application might have a source file with the following lines:

```
#define WINDOWS         /* Will be automatically defined if your
                           compiler defines _WINDOWS */
#define WIN32
#include <sccex.h>
```

The developer's application should be linked to the product DLLs through the provided libraries.

4.3.2 Options and Information Storage

When using the Export products, a list of available filters and a list of available display engines are built by the technology, usually the first time the product runs. You do not need to ship these lists with your application. The lists are automatically recreated if corrupted or deleted.

The files used to store this information are stored in an .oit subdirectory in \Documents and Settings*user name*\Application Data.

If an .oit directory does not exist in the user's directory, the directory is created automatically. The files are automatically regenerated if corrupted or deleted.

The files are:

- *.d = Display Engine lists
- *.opt = Persistent options

Some applications and services may run under a local system account for which there is no users "application data" folder. The technology first does a check for an environment variable called OIT_DATA_PATH. Then it checks for APPDATA, and then LOCALAPPDATA. If none of those exist, the options files are put into the executable path of the UT module.

These file names are intended to be unique enough to avoid conflict for any combination of machine name and install directory. This allows the user to run products in separate directories without having to reload the files above. The file names are built from an 11-character string derived from the directory the Oracle Outside In technology resides in and the name of the machine it is being run on. The string is generated by code derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

The software still functions if these lists cannot be created for some reason. In that situation, however, significant performance degradation should be expected.

4.3.3 Structure Alignment

Oracle Outside In is built with 8-byte structure alignment. This is the default setting for most Windows compilers. This and other compiler options that should be used are demonstrated in the files provided with the sample applications in samples\win.

4.3.4 Character Sets

The strings passed in the Windows API are Microsoft Code Page 1252 by default.

To optimize performance on systems that do not require DBCS support, a second character mapping bin file, that does not contain any of the DBCS pages, is now included. The second bin file gives additional performance benefits for English documents, but cannot handle DBCS documents. To use the new bin file, replace the cmmap000.bin with the new bin file, cmmap000.sbc. For clarity, a copy of the cmmap000.bin file (cmmap000.dbc) is also included. Both cmmap000.sbc and cmmap000.dbc are located in the \common directory of the technology.

4.3.5 Runtime Considerations

The files used by the product must be in the same directory as the developer's executable.

4.4 Default Font Aliases

The technology includes the following default font alias map for Windows. The first value is the original font, the second is the alias.

- Chicago = Arial
- Geneva = Arial
- New York = Times New Roman
- Helvetica = Arial
- Helv = Arial
- times = Times New Roman
- Times = Times New Roman
- Tms Roman = Times New Roman
- itc zapfdingbats = Zapfdinbats
- itc zapf dingbats = Zapfdinbats

4.5 Changing Resources

Oracle Outside In Web View Export ships with the necessary files for OEMs to change any of the strings in the technology as they see fit.

Strings are stored in the lodlgstr.h file found in the resource directory. The file can be edited using any text editor.

> **Note:** Do not directly edit the scclo.rc file. Strings are saved with their identifiers in lodlgstr.h. If a new scclo.rc file is saved, it will contain numeric identifiers for strings, instead of their #define'd names.

Once the changes have been made, the updated scclo.dll file can be rebuilt using the following steps:

1. Compile the .res file:

```
rc /fo ".\scclo.res" /i "<path to header (.h) files folder>" /d "NDEBUG"
```

```
scclo.rc
```

2. Link the scclo.res file you've created with the scclo.obj file found in the resource directory to create a new scclo.dll:

```
link /DLL /OUT:scclo.dll scclo.obj scclo.res
```

> **Note:** Developers should make sure they have set up their environment variables to build the library for their specific architecture. For Windows x86_32, when compiling with VS 2005, the solution is to run vsvars32.bat (in a standard VS 2005 installation, this is found in C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\). If this works correctly, you will see the statement, "Setting environment for using Microsoft Visual Studio 2005 x86 tools." If you do not complete this step, you may have conflicts that lead to unresolved symbols due to conflicts with the Microsoft CRT.

3. Embed the manifest (which is created in the \resource directory during step 2) into the new DLL:

```
mt -manifest scclo.dll.manifest -outputresource:scclo.dll;2
```

If you are not using Microsoft Visual Studio, substitute the appropriate development tools from your enviroment.

> **Note:** In previous versions of Oracle Outside In, it was possible to directly edit the SCCLO.DLL using Microsoft Visual Studio. Oracle Outside In DLLs are now digitally signed. Editing the signed DLL is not advisable.

5 UNIX Implementation Details

The UNIX implementation of the Export product set is delivered as a set of shared libraries. For a list of the currently supported platforms, see:

http://www.oracle.com/technetwork/indexes/documentation/index.html#middleware

Click on Outside In Technology. On the Getting Started tab of the Outside In library, click on the current release number under the Certified Platforms heading.

This chapter includes the following sections:

5.1 Installation

To install the demo version of the SDK, copy the tgz file corresponding to your platform (available on the web site) to a local directory of your choice. Decompress the tgz file and then extract from the resulting tar file as follows:

```
gunzip tgzfile
tar xvf tarfile
```

The installation directory should contain the following directory structure.

Directory	Description
/redist	Contains a working copy of the UNIX version of the technology.
/sdk/assets	Contains Javascript, CSS, and image files that are used by the Web View Export output.
/sdk/common	Contains the C include files needed to build or rebuild the technology.

Directory	Description
/sdk/demo	Contains the compiled executables of the sample applications.
/sdk/resource	Contains localization resource files. For more details, see Section 5.8, "Changing Resources."
/sdk/samplecode	Contains a subdirectory holding the source code for a sample application. For more details, see Chapter 3, "Sample Applications."
/sdk/samplefiles	Contains sample files designed to exercise the technology.

5.1.1 NSF Support

Notes Storage Format (NSF) files are produced by the Lotus Notes Client or the Lotus Domino server. The NSF filter is the only Outside In filter that requires the native application to be present to filter the input documents. Due to integration with an outside application, NSF support will not work with redirected I/O nor will it work when an NSF file is embedded in another file. Lotus Domino version 8 must be installed on the same machine as OIT. The NSF filter is currently only supported on the Win32, Win x86-64, Linux x86-32, and Solaris Sparc 32 platforms. SCCOPT_LOTUSNOTESDIRECTORY is a Windows-only option and is ignored on Unix.

Additional steps must be taken to prepare the system. It is necessary to know the name of the directory in which Lotus Domino has been installed. On Linux, this default directory is /opt/ibm/lotus/notes/latest/linux. On Solaris, it is /opt/ibm/lotus/notes/latest/sunspa.

- In the Lotus Domino directory, check for the existence of a file called "notes.ini". If the file "notes.ini" does not exist, create it in that directory and ensure that it contains the following single line:

  [Notes]

- Add the Lotus Domino directory to the $LD_LIBRARY_PATH environment variable.
- Set the environment variable $Notes_ExecDirectory to the Lotus Domino directory.

5.2 Libraries and Structure

On UNIX platforms the Oracle Outside In products are delivered with a set of shared libraries. All libraries should be installed to a single directory. Depending upon your application, you may also need to add that directory to the system's runtime search path. For more details, see Section 5.6, "Environment Variables."

The following is a brief description of the included libraries and support files. In instances where a file extension is listed as .*, the file extension varies for each UNIX platform (**sl** on HP-UX, **so** on Linux and Solaris).

5.2.1 API Libraries

These libraries implement the API. They should be linked with the developer's application.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libsc_da.*	Data Access module	X	X	X	X	X	X
libsc_ex.*	Export module	X	X	X	X	X	X
libsc_fi.*	File Identification module (identifies files based on their contents).	X	X	X	X	X	X

The File ID Specification may not be used directly by any application or workflow without it being separately licensed expressly for that purpose.

5.2.2 Support Libraries

The following libraries are used for support.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libccflex.*	A data model adapter that converts from stream model utilized by Outside In filters to the FlexionDoc Tree model used as a basis by XML Export.					X	
libexpatw.*	A third-party XML parser.					X	
liboc_emul.*	Output component emulation module	X	X	X	X	X	X
libos_gd.*	The internal rendering GDI implementation. **32-bit Linux and Solaris SPARC only.**	X	X		X	X	
libos_pdf.*	PDF generation module			X			
libos_xwin.*	The native GDI implementation	X	X		X	X	
libsc_anno.*	The annotation module	X	X	X			X
libsc_ca.*	Content Access module (provides organized chunker data for the developer)	X	X	X			X
libsc_ch.*	Chunker (provides caching of and access to filter data for the export engines)	X	X	X	X	X	X
libsc_du.*	Display Utilities module (includes text formatting)	X	X	X	X	X	X
libsc_fmt.*	Formatting module (resolves numbers to formatted strings)	X	X	X	X	X	X
libsc_fut.*	Filter utility module	X	X	X	X	X	X
libsc_ind.*	Indexing engine. In Search Export, it handles common functionality.	X	X	X	X		X
libsc_lo.*	Localization library (all strings, menus, dialogs and dialog procedures reside here)	X	X	X	X	X	X
libsc_sd.*	Schema Definition Module Manager (brokers multiple Schema Definition Modules)					X	
libsc_ut.*	Utility functions, including IO subsystem	X	X	X	X	X	X

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libsc_xp.*	XPrinter bridge	X	X		X	X	
libsdflex.*	Schema Definition module (handles conversion of XML string names and attribute values to compact binary representations and vice versa)					X	
libwv_core.*	The Abstraction layer	X	X	X	X	X	X
libwv_gdlib.so	The GDI rendering module. **32-bit Linux and Solaris SPARC only.**	X	X		X	X	

5.2.3 Engine Libraries

The following libraries are used for display purposes.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libde_bmp.*	Raster rendering engine (TIFF, GIF, BMP, PNG, PCX…)			X		X	X
libde_vect.*	Vector/Presentation rendering engine (PowerPoint, Impress, Freelance)	X	X	X		X	X
libde_ss.*	Spreadsheet/Database (Excel, Calc, Lotus 123)		X	X		X	
libde_tree*	Archive (ZIP, GZIP, TAR…)		X	X			
libde_wp.*	Document (Word, Writer, WordPerfect)		X	X	X		X

5.2.4 Filter and Export Filter Libraries

The following libraries are used for filtering.

libex_gdsf must be linked with libsc_img.* at compile time. This forces the filter to be dependent on libsc_img.* at runtime, even though that module may not be used directly. If you want to reduce your application's physical footprint, you can experiment with unlinking libsc_img.*.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libvs_*.*	Filters for specific file types (there are more than 150 of these filters, covering more than 600 file formats)	X	X	X	X	X	X
libex_gdsf.*	Export filter for GIF, JPEG, and PNG graphics files	X				X	X
libex_h5.*	Export filter for HTML5 files						X
libsc_img.*	Image conversion module	X	X			X	X
libex_itext.*	Export filter for SearchText				X		
libex_html.*	Export filter for HTML files	X					
libex_img.*	Extended image conversion module		X				
libex_xml.*	Export filter for XML files using the Flexiondoc schema					X	

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libex_page.*	Export filter for XML files using the PageML schema				X		
libex_ pagelayout.*	Page Layout module			X			
libex_ixml.*	Export filters for XML files using the SearchML schema				X		
libex_ihtml.*	Export filter for SearchHTML				X		

5.2.5 Premier Graphics Filters

The following are graphics filters.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
libi*.*	These files are the import filters for premier graphics formats.	X	X	X	X	X	X
libis_unx2.*	Interface to premier graphics filters	X	X	X	X	X	X

5.2.6 Additional Files

The following files are also used.

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
adinit.dat	Support file for the vsacad and vsacd2 filters	X	X	X	X	X	X
ccbf.so	Internal				X		
cmmap000.bin	Tables for character mapping (all character sets)	X	X	X	X	X	X
cmmap000.sbc	Tables for character mapping (single-byte character sets). This file is located in the /common directory.	X	X	X	X	X	X
cmmap000.dbc	Identical to cmmap000.bin, but renamed for clarity (.dbc = double-byte character). This file is located in the common directory.	X	X	X	X	X	X
exbf.so	Internal				X		

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
flexiondoc.dtd	The DTD version of the Flexiondoc schema					X	
flexiondoc.xsd	The schema version of the Flexiondoc schema					X	
libfreetype.so.6	TrueType font rendering module for the GD output solution. **32-bit Linux and Solaris Sparc only.**	X	X	X	X	X	
oitnsf.id	Support file for the vsnsf filter.	X	X	X	X	X	X
pageml.dtd	The Document Type Definition for the PageML schema				X		
pageml.xsd	The Extensible Schema Definition for the PageML schema				X		
searchml3.dtd	The Document Type Definitions for the SearchML schema				X		
searchml3.xsd	The Extensible Schema Definitions for the SearchML schema				X		

5.3 The Basics

Sample applications are provided with the SDK. These applications demonstrate most of the concepts described in this manual. For a complete description of the sample applications, see Chapter 3, "Sample Applications."

5.3.1 What You Need in Your Source Code

Any source code that uses this product should `#include` the file sccex.h and `#define` UNIX. For example, a 32-bit UNIX application might have a source file with the following lines:

```
#define UNIX
#include <sccex.h>
```

and a 64-bit UNIX application might have a source file with the following lines:

```
#define UNIX
#define UNIX_64
#include <sccex.h>
```

5.3.2 Information Storage

This software is based on the Oracle Outside In Viewer Technology (or simply "Viewer Technology"). A file of default options is always created, and a list of available filters

and a list of available display engines are also built by the technology, usually the first time the product runs (for UNIX implementations). You do not need to ship these lists with your application.

Lists are stored in the $HOME/.oit directory. If the $HOME environment variable is not set, the files are put in the same directory as the Oracle Outside In Technology. If a /.oit directory does not exist in the user's $HOME directory, the .oit directory is created automatically by the technology. The files are automatically regenerated if corrupted or deleted.

The files are:

- *.d: Display engine list
- *.opt: Persistent options

The filenames are intended to be unique enough to avoid conflict for any combination of machine name and install directory. This is intended to prevent problems with version conflicts when multiple versions of the Oracle Outside In Technology and/or other Oracle Outside In Technology-based products are installed on a single system. The filenames are built from an 11-character string derived from the directory the Oracle Outside In Technology resides in and the name of the machine it is being run on. The string is generated by code derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

The products still function if these files cannot be created for some reason. In that situation, however, significant performance degradation should be expected.

5.4 Character Sets

The strings passed in the UNIX API are ISO8859-1 by default.

To optimize performance on systems that do not require DBCS support, a second character mapping bin file, that does not contain any of the DBCS pages, is included. The second bin file gives additional performance benefits for English documents, but cannot handle DBCS documents. To use the new bin file, replace the cmmap000.bin with the new bin file, cmmap000.sbc. For clarity, a copy of the cmmap000.bin file (cmmap000.dbc) is also included. Both cmmap000.sbc and cmmap000.dbc are located in the /sdk/common directory of the technology.

5.5 Runtime Considerations

The following is information to consider during run-time.

5.5.1 OLE2 Objects

Some documents that the developer is attempting to convert may contain embedded OLE2 objects. There are platform-dependent limits on what the technology can do with OLE2 objects. However, Oracle Outside In attempts to take advantage of the fact that some documents accompany an OLE2 embedding with a graphic "snapshot," in the form of a Windows metafile.

On all platforms, when a metafile snapshot is available, the technology uses it to convert the object. When a metafile snapshot is not available on UNIX platforms, the technology is unable to convert the OLE2 object.

5.5.2 Signal Handling

These products trap and handle the following signals:

- SIGABRT
- SIGBUS
- SIGFPE
- SIGILL
- SIGINT
- SIGSEGV
- SIGTERM

Developers who wish to override our default handling of these signals should set up their own signal handlers. This may be safely done after the developer's application has called DAInitEx().

Note: The Java Native Interface (JNI) allows Java code to call and be called by native code (C/C++ in the case of OIT). You may run into problems if Java isn't allowed to handle signals and forward them to OIT. If OIT catches the signals and forwards them to Java, the JVMs will sometimes crash. OIT installs signal handlers when DAInitEx() is called, so if you call OIT after the JVM is created, you will need to use libjsig. Refer here for more information:

http://www.oracle.com/technetwork/java/javase/index-137495.html

5.5.3 Runtime Search Path and $ORIGIN

Libraries and sample applications are all built with the $ORIGIN variable as part of the binaries' runtime search path. This means that at runtime, OIT libraries will automatically look in the directory they were loaded from to find their dependent libraries. You don't necessarily need to include the technology directory in your LD_LIBRARY_PATH or SHLIB_PATH.

As an example, an application that resides in the same directory as the OIT libraries and includes $ORIGIN in its runtime search path will have its dependent OIT libraries found automatically. You will still need to include the technology directory in your linker's search path at link time using something like -L and possibly -rpath-link.

Another example is an application that loads OIT libraries from a known directory. The loading of the first OIT library will locate the dependent libraries.

Note: This feature does not work on AIX and FreeBSD.

5.6 Environment Variables

Several environment variables may be used at run time. Following is a short summary of those variables and their usage.

Variable	Description
$LD_LIBRARY_PATH (FreeBSD, HP-UX Itanium 64, Linux, Solaris) $SHLIB_PATH (HP-UX PA-RISC 32) $LIBPATH (AIX, iSeries)	These variables help your system's dynamic loader locate objects at runtime. If you have problems with libraries failing to load, try adding the path to the Oracle Outside In libraries to the appropriate environment variable. See your system's manual for the dynamic loader and its configuration for details. Note that for products that have a 64-bit PA-RISC, 64-bit Solaris and Linux PPC/PPC64 distributable, they will also go under $LD_LIBRARY_PATH.
$HOME	Must be set to allow the system to write the option, filter and display engine lists. For details, see Section 5.3.2, "Information Storage."

5.7 Default Font Aliases

Outside In Technology (OIT) will use the fonts installed on the host system. If a file being converted with OIT uses fonts that are available on the host system, no substitution should occur and the original font from the input file will be used. If the original font used in the input file is not available on the host system, then OIT will first check to see if an alias has been set for the font using the SCCOPT_PRINTFONTALIAS option (see documentation for details on using this option). If there is an alias available, and the alias font is available on the host system, then OIT will use this font. If no alias is set or the alias font is not available on the host system, then a substitution will occur. The first attempt at a substitution will use the default font specified by the SCCOPT_DEFAULTPRINTFONT option. If this font has the glyphs to be rendered, it will be used ahead of all other potential substitutions. In some cases, the default font cannot be used because it does not contain the glyphs required to render the text from the input file.

For example, a default font of Arial may not contain glyphs required to render certain Asian languages. In these cases, OIT will pick another font that does have the glyphs, if one exists. The mechanism for picking that other font is not very predictable, and often leads to bad results (picking a serifed font for a non-serifed, variable width for a fixed width, etc.). Therefore, the best solution for users is to have as many fonts available to OIT as possible. This will avoid font substitutions and provide the most accurate rendering of the original file.

Note that font substitutions can lead to different wrapping, which can lead to different numbers of pages rendered by OIT versus the native application. This further underscores the importance of a host system with as many fonts as possible.

The technology includes the following default font alias map for UNIX platforms. The first value is the original font, and the second is the alias.

- 61 = Liberation Sans
- Andale Mono = Liberation Sans
- Courier = Liberation Sans
- Courier New = Liberation Sans
- Lucida Console = Liberation Sans
- MS Gothic = Liberation Sans
- MS Mincho = Liberation Sans

- OCR A Extended = Liberation Sans
- OCR B = Liberation Sans
- Agency FB = Liberation Sans
- Arial = Liberation Sans
- Arial Black = Liberation Sans
- Arial Narrow = Liberation Sans
- Arial Rounded MT = Liberation Sans
- Arial Unicode MS = Liberation Sans
- Berline Sans FB = Liberation Sans
- Calibri = Liberation Sans
- Frank Gothic Demi = Liberation Sans
- Frank Gothic Medium Cond = Liberation Sans
- Franklin Gothic Book = Liberation Sans
- Futura = Liberation Sans
- Geneva = Liberation Sans
- Gill Sans = Liberation Sans
- Gill Sans MT = Liberation Sans
- Lucida Sans Regular = Liberation Sans
- Lucida Sans Unicode = Liberation Sans
- Modern No. 20 = Liberation Sans
- Tahoma = Liberation Sans
- Trebuchet MS = Liberation Sans
- Tw Cen MT = Liberation Sans
- Verdana = Liberation Sans
- Albany = Liberation Sans
- Franklin Gothic = Liberation Sans
- Franklin Demi = Liberation Sans
- Franklin Demi Cond = Liberation Sans
- Franklin Gothic Heavy = Liberation Sans
- Algerian = Liberation Serif
- Baskerville = Liberation Serif
- Bell MT = Liberation Serif
- Bodoni MT = Liberation Serif
- Bodoni MT Black = Liberation Serif
- Book Antiqua = Liberation Serif
- Bookman Old Style = Liberation Serif
- Calisto MT = Liberation Serif

- Cambria = Liberation Serif
- Centaur = Liberation Serif
- Century = Liberation Serif
- Century Gothic = Liberation Serif
- Century Schoolbook = Liberation Serif
- Elephant = Liberation Serif
- Footlight MT Light = Liberation Serif
- Garamond = Liberation Serif
- Georgia = Liberation Serif
- Goudy Old Style = Liberation Serif
- Lucida Bright = Liberation Serif
- MS Serif = Liberation Serif
- New York = Liberation Serif
- Palatino = Liberation Serif
- Perpetua = Liberation Serif
- Times = Liberation Serif
- times = Liberation Serif
- Times New Roman = Liberation Serif

5.8 Changing Resources

All of the strings used in the UNIX versions of Oracle Outside In products are contained in the lodlgstr.h file. This file, located in the resource directory, can be modified for internationalization and other purposes. Everything necessary to rebuild the resource library to use the modified source file is included with the SDK.

In addition to lodlgstr.h, the scclo.o object file is provided. This is necessary for the linking phase of the build. A makefile has also been provided for building the library. The makefile allows building on all of the UNIX platforms supported by Oracle Outside In. It may be necessary to make minor modifications to the makefile so the system header files and libraries can be found for compiling and linking.

Standard INCLUDE and LIB *make* variables are defined for each platform in the makefile. Edit these variables to point to the header files and libraries on your particular system. Other make variables are:

- TECHINCLUDE: May need to be edited to point to the location of the Oracle Outside In /common header files supplied with the SDK.
- BUILDDIR: May need to be edited to point to the location of the makefile, lodlgstr.h, and scclo.o (which should all be in the same directory).

After these variables are set, change to the build directory and type make. The libsc_lo resource library is built and placed in the appropriate platform-specific directory. To use this library, copy it into the directory where the Oracle Outside In product is stored and the new, modified resource strings are used by the technology.

Menu constants are included in lomenu.h in the common directory.

5.9 Linux Compiling and Linking

This section discusses issues involving Linux compiling and linking.

5.9.1 Library Compatibility

This section discusses Linux compatibility issues when using libraries.

5.9.1.1 GLIBC and Compiler Versions

The following table indicates the compiler version used and the minimum required version of the GNU standard C library needed for Oracle Outside In operation.

Distribution	Compiler Version	GLIBC Version
x64 Linux	4.1.2	libc.so.6

5.9.1.2 Other Libraries

In addition to libc.so.6, Oracle Outside In is dependent upon the following libraries:

- libstdc++.so.6
- libgcc_so.1

The following table summarizes what is included with the RedHat and SUSE distributions supported by Oracle Outside In and what needs to be added/modified to make Oracle Outside In run on these systems.

5.9.1.2.1 Libraries on Linux Systems as Distributed (IA32)

SUSE 9.0

Included	To be added
libc.so.6 version	/lib/libc.so.6 (GLIBC 2.3.4)
libstdc++	/usr/lib/libstdc++.so.5.0.6 + old libraries
libgcc_s.so.1	/lib/libgcc_s.so.1
Required to Use Oracle Outside In	■ Default system install has proper libstdc++.so.5 ■ Default system install has libgcc_so.1 ■ Update to >= libXm.so.3.0.2 (OpenMotif >=2.2.3) ■ Install X libraries

5.9.2 Compiling and Linking

The libsc_ex.so and libsc_da.so are the only libraries that must be linked with your applications. They can be loaded when your application starts by linking them directly at compile time or they can be loaded dynamically by your application using library load functions (for example, dlopen).

The following are example command lines used to compile the sample application **exsimple** from the /sdk/samplecode directory. This command line is only an example. The actual command line required on the developer's system may vary.

The example assumes that the include and library file search paths for the technology libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L

library file path options, respectively, so the compiler and linker can locate all required files.

5.9.2.1 Linux 64-bit

```
gcc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c
-I/usr/local/include -I../../common -L../../demo -L/usr/local/lib -lsc_ex -lsc_da
-DUNIX_64 -Wl,-rpath,../../demo -Wl,-rpath,'${ORIGIN}'
```

6

Data Access Common Functions

The Data Access module is common to all Oracle Outside In SDKs. It provides a way to open a generic handle to a source file. This handle can then be used in the functions described in this chapter.

This chapter includes the following sections:

6.1 DAInitEx

This function tells the Data Access module to perform any necessary initialization it needs to prepare for document access. This function must be called before the first time the application

uses the module to retrieve data from any document. This function supersedes the old DAInit and DAThreadInit functions.

> **Note:** DAInitEx should only be called once per application, at application startup time. Any number of documents can be opened for access between calls to DAInitEx and DADeInit. If DAInitEx succeeds, DADeInit must be called regardless of any other API calls.

If the ThreadOption parameter is set to something other than DATHREAD_INIT_NOTHREADS, then this function's preparation includes setting up mutex function pointers to prevent threads from clashing in critical sections of the technology's code. The developer must actually code the threads after this function has been called. DAInitEx should be called only once per process and should be called before the developer's application begins the thread.

> **Note:** Multiple threads are supported for all Windows platforms and the 32-bit versions of Linux x86 and Solaris SPARC. Failed initialization of the threading function will not impair other API calls. If threading isn't initialized or fails, stub functions are called instead of mutex functions.

Prototype

```
DAERR DAInitEx(VTSHORT ThreadOption, VTDWORD dwFlags);
```

Parameters

- ThreadOption: can be one of the following values:
  - DATHREAD_INIT_NOTHREADS: No thread support requested.
  - DATHREAD_INIT_PTHREADS: Support for PTHREADS requested.
  - DATHREAD_INIT_NATIVETHREADS: Support for native threading requested. Supported only on Microsoft Windows platforms and Oracle Solaris.
- dwFlags: can be one or more of the following flags OR-ed together
  - OI_INIT_DEFAULT: Options Load and Save are performed normally
  - OI_INIT_NOSAVEOPTIONS: The options file will not be saved on exit
  - OI_INIT_NOLOADOPTIONS: The options file will not be read during initialization.

Return Values

- DAERR_OK: If the initialization was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.2 DADeInit

This function tells the Data Access module that it will not be asked to read additional documents, so it should perform any cleanup tasks that may be necessary. This function should be called at application shutdown time, and only if the module was successfully initialized with a call to DAInitEx.

Prototype

```
DAERR DADeInit();
```

Return Values

- DAERR_OK: If the de-initialization was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.3 DAOpenDocument

Opens a source file to make it accessible by one or more of the data access technologies. If DAOpenDocument succeeds, DACloseDocument must be called regardless of any other API calls.

For IO types other than IOTYPE_REDIRECT, the subdocument specification may be specified as part of the file's path. This is accomplished by appending a question mark delimiter to the path, followed by the subdocument specification. For example, to specify the third item within the file c:\docs\file.zip, specify the path c:\docs\file.zip?item.3 in the call to DAOpenDocument. DAOpenDocument always attempts to open the specification as a file first. In the unlikely event there is a file with the same name (including the question mark) as a file plus the subdocument specification, that file is opened instead of the archive item.

To take advantage of this feature when providing access to the input file using redirected IO, a subdocument specification must be provided via a response to an IOGetInfo message, IOGETINFO_SUBDOC_SPEC. To specify an item in an archive, first follow the standard redirected IO methods to provide a BASEIO pointer to the archive file itself. To specify an item within the archive, a redirected IO object must respond to the IOGETINFO_SUBDOC_SPEC message by copying to the supplied buffer the subdocument specification of the archive item to be opened. This message is received during the processing of DAOpenDocument.

Prototype

```
DAERR DAOpenDocument(
   VTLPHDOC   lphDoc,
   VTDWORD    dwSpecType,
   VTLPVOID   pSpec,
   VTDWORD    dwFlags);
```

Parameters

- lphDoc: Pointer to a handle that will be filled with a value uniquely identifying the document to data access. The developer uses this handle in subsequent calls to data access to identify this particular source file. This is not an operating system file handle.
- dwSpecType: Describes the contents of pSpec. Together, dwSpecType and pSpec describe the location of the source file. Must be one of the following values:
  - IOTYPE_ANSIPATH: Windows only. pSpec points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) file name conventions.
  - IOTYPE_UNICODEPATH: Windows only. pSpec points to a NULL-terminated full path name using the Unicode character set and NTFS (Win32 and Win64) file name conventions.
  - IOTYPE_UNIXPATH: UNIX platforms only. pSpec points to a NULL-terminated full path name using the system default character set and UNIX path conventions. Unicode paths can be accessed on UNIX platforms by using a UTF-8 encoded path with IOTYPE_UNIXPATH.
  - IOTYPE_REDIRECT: All platforms. pSpec points to a developer-defined struct that allows the developer to redirect the IO routines used to read the file. For more information, see Chapter 8, "Redirected IO."

  - IOTYPE_ARCHIVEOBJECT: All platforms. Opens an embedded archive object for data access. pSpec points to a structure IOSPECARCHIVEOBJECT (see Section 6.3.2, "IOSPECARCHIVEOBJECT Structure" for details) that has been filled with values returned in a SCCCA_OBJECT content entry from Content Access.
  - IOTYPE_LINKEDOBJECT: All platforms. Opens an object specified by a linked object for data access. pSpec points to a structure IOSPECLINKEDOBJECT (see Section 6.3.1, "IOSPECLINKEDOBJECT Structure") that has been filled with values returned in an SCCCA_BEGINTAG or SCCCA_ENDTAG with a subtype of SCCCA_LINKEDOBJECT content entry from Content Access.

- pSpec: File location specification.
- dwFlags: The low WORD is the file ID for the document (0 by default). If you set the file ID incorrectly, the technology fails. If set to 0, the file identification technology determines the input file type automatically. The high WORD should be set to 0.

Return Values

- DAERR_OK: Returned if the open was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.3.1 IOSPECLINKEDOBJECT Structure

Structure used by DAOpenDocument.

Prototype

```
typedef struct IOSPECLINKEDOBJECTtag
   {
   VTDWORD    dwStructSize;
   VTSYSPARAM hDoc;
   VTDWORD    dwObjectId;  /* Object identifier. */
   VTDWORD    dwType;      /* Linked Object type */
                           /* (SO_LOCATORTYPE_*) */
   VTDWORD    dwParam1;    /* parameter for DoSpecial call */
   VTDWORD    dwParam2;    /* parameter for DoSpecial call */
   VTDWORD    dwReserved1; /* Reserved. */
   VTDWORD    dwReserved2; /* Reserved. */
} IOSPECLINKEDOBJECT,       * PIOSPECLINKEDOBJECT;
```

6.3.2 IOSPECARCHIVEOBJECT Structure

Structure used by DAOpenDocument.

Prototype

```
typedef struct IOSPECARCHIVEOBJECTtag
   {
   VTDWORD dwStructSize;
   VTDWORD hDoc;        /* Parent Doc hDoc */
   VTDWORD dwNodeId;    /* Node ID */
   VTDWORD dwStreamId;
   VTDWORD dwReserved1; /* Must always be 0 */
   VTDWORD dwReserved2; /* Must always be 0 */
} IOSPECARCHIVEOBJECT,   * PIOSPECARCHIVEOBJECT;
```

6.4 DACloseDocument

This function is called to close a file opened by the reader that has not encountered a fatal error.

Prototype

```
DAERR DACloseDocument(
   VTHDOC hDoc);
```

Parameters

- hDoc: Identifier of open document. Must be a handle returned by the DAOpenDocument function.

Return Value

- DAERR_OK: Returned if close succeeded. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.5 DARetrieveDocHandle

This function returns the document handle associated with any type of Data Access handle. This allows the developer to only keep the value of hItem, instead of both hItem and hDoc.

Prototype

```
DAERR DARetrieveDocHandle(
   VTHDOC     hItem,
   VTLPHDOC   phDoc);
```

Parameters

- hItem: Identifier of open document. May be the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions in the data access submodule. Passing in an hDoc created by DAOpenDocument for this parameter results in an error.
- phDoc: Pointer to a handle to be filled with the document handle associated with the passed subhandle.

Return Value

- DAERR_OK: Returned if the handle in phDoc is valid. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.6 DASetOption

This function is called to set the value of a data access option.

Prototype

```
DAERR DASetOption(
   VTHDOC      hDoc,
   VTDWORD     dwOptionId,
   VTLPVOID    pValue,
   VTDWORD     dwValueSize);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.). Setting an option for a VTHDOC affects all subhandles opened under it, while setting an option for a subhandle affects only that handle.

If this parameter is NULL, then setting the option affects all documents opened thereafter. Once an option is set using the NULL handle, this option becomes the default option thereafter. This parameter should only be set to NULL if the option being set can take that value.

- dwOptionId: The identifier of the option to be set.
- pValue: Pointer to a buffer containing the value of the option.
- dwValueSize: The size in bytes of the data pointed to by pValue. For a string value, the NULL terminator should be included when calculating dwValueSize.

Return Value

- DAERR_OK: Returned if DASetOption succeeded. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.7 DAGetOption

This function is called to retrieve the value of a data access option. The results of a call to this option are only valid if DASetOption has already been called on the option.

Prototype

```
DAERR DAGetOption(
   VTHDOC    hItem,
   VTDWORD   dwOptionId,
   VTLPVOID  pValue,
   VTLPDWORD pSize);
```

Parameters

- hItem: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.). Getting an option for a VTHDOC gets the value of that option for that handle, which may be different than the subhandle's value.
- dwOptionId: The identifier of the option to be returned.
- pValue: Pointer to a buffer containing the value of the option.
- pSize: This VTDWORD should be initialized by the caller to the size of the buffer pointed to by pValue. If this size is sufficient, the option value is copied into pValue and pSize is set to the actual size of the option value. If the size is not sufficient, pSize is set to the size of the buffer needed for the option and an error is returned.

Return Value

- DAERR_OK: Returned if DAGetOption was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.8 DAGetFileId

This function allows the developer to retrieve the format of the file based on the technology's content-based file identification process. This can be used to make intelligent decisions about how to process the file and to give the user feedback about the format of the file they are working with.

Note: In cases where File ID returns a value of FI_UNKNOWN, this function will apply the Fallback Format before returning a result.

Prototype

```
DAERR DAGetFileId(
   VTHDOC      hDoc,
   VTLPDWORD   pdwFileId);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, a VTHEXPORT returned by the EXOpenExport function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHEXPORT, VTHCONTENT, VTHTEXT, etc.).
- pdwFileId: Pointer to a DWORD that receives a file identification number for the file. These numbers are defined in sccfi.h.

Return Value

- DAERR_OK: Returned if DAGetFileId was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.9 DAGetFileIdEx

This function allows the developer to retrieve the format of the file based on the technology's content-based file identification process. This can be used to make intelligent decisions about how to process the file and to give the user feedback about the format of the file they are working with. This function has all the functionality of DAGetFileID and adds the ability to return the raw FI value; in other words, the value returned by normal FI, without applying the FallbackFI setting.

Prototype

```
DAERR DAGetFileIdEx(
   VTHDOC      hDoc,
   VTLPDWORD   pdwFileId,
   VTDWORD     dwFlags);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHEXPORT, VTHCONTENT, VTHTEXT, etc.).
- pdwFileId: Pointer to a DWORD that receives a file identification number for the file. These numbers are defined in sccfi.h.
- dwFlags: DWORD that allows user to request specific behavior.
  - DA_FILEINFO_RAWFI: This flag tells DAGetFileIdEx() to return the result of the File Identification operation before Extended File Ident. is performed and without applying the FallbackFI value.

Return Value

- DAERR_OK: Returned if DAGetFileIdEx was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned. See the following tables for examples of expected output depending on the value of various options.

Values with RAWFI turned off

Input file type	ExtendedFI	FallbackID	DAGetFileId	DAGetFileIdEx
true binary	off	fallback value	fallback value	fallback value
true binary	on	fallback value	fallback value	fallback value
true text	off	fallback value	fallback value	fallback value
true text	on	fallback value	40XX	40XX

Values with RAWFI turned on

Input file type	ExtendedFI	FallbackID	DAGetFileId	DAGetFileIdEx
true binary	off	fallback value	fallback value	1999
true binary	on	fallback value	fallback value	1999
true text	off	fallback value	fallback value	1999
true text	on	fallback value	40XX	1999

6.10 DAGetErrorString

This function returns to the developer a string describing the input error code. If the error string returned does not fit the buffer provided, it is truncated.

```
VTVOID DAGetErrorString(
   DAERR      deError,
   VTLPVOID   pBuffer,
   VTDWORD    dwBufSize);
```

Parameters

- Error: Error code passed in by the developer for which an error message is to be returned.
- pBuffer: This buffer is allocated by the caller and is filled in with the error message by this routine. The error message will be a NULL-terminated string.
- dwBufSize: Size of what pBuffer points to in bytes.

Return Value

- none

6.11 DAGetTreeCount

This function is called to retrieve the number of records in an archive file.

```
DAERR DAGetTreeCount(
      VTHDOC      hDoc,
      VTLPDWORD   lpRecordCount);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by any of the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.).
- lpRecordCount: A pointer to a VTLPDWORD that is filled with the number of stored archive records.

Return Value

- DAERR_OK: DAGetTreeCount was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: The selected file does not contain an archive section, or the requested record does not exist.

6.12 DAGetTreeRecord

This function is called to retrieve information about a record in an archive file.

```
DAERR DAGetTreeRecord(
      VTHDOC         hDoc,
      PSCCDATREENODE pTreeNode);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle by any of the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.).
- pTreeNode: A pointer to a PSCCDATREENODE structure that is filled with information about the selected record.

Return Values

- DAERR_OK: DAGetTreeRecord was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: The selected file does not contain an archive section, or the requested record does not exist.
- DAERR_EMPTYFILE: Empty file.
- DAERR_PROTECTEDFILE: Password protected or encrypted file.
- DAERR_SUPFILEOPENFAILS: Supplementary file open failed.
- DAERR_FILTERNOTAVAIL: The file's type is known, but the appropriate filter is not available.
- DAERR_FILTERLOADFAILED: An error occurred during the initialization of the appropriate filter.

6.12.1 SCCDATREENODE Structure

This structure is passed by the OEM through the DAGetTreeRecord function. The structure is defined in sccda as follows:

```
typedef struct SCCDATREENODEtag{
   VTDWORD   dwSize;
   VTDWORD   dwNode;
   VTBYTE    szName[1024];
   VTDWORD   dwFileSize;
   VTDWORD   dwTime;
   VTDWORD   dwFlags;
   VTDWORD   dwCharSet;
   } SCCDATREENODE,   *PSCCDATREENODE;
```

Parameters

- dwSize: Must be set by the OEM to sizeof(SCCDATREENODE).

- dwNode: The number of the record for which information is being retrieved. The first node is node 0.
- szName: A buffer to hold the name of the record.
- dwFileSize: Returns the file size, in bytes, of the requested record.
- dwTime: Returns the timestamp of the requested record, in MS-DOS time.
- dwFlags: Returns additional information about the node. It can be a combination of the following:
  - SCCDA_TREENODEFLAG_FOLDER: Indicating that the selected node is a folder and not a file.
  - SCCDA_TREENODEFLAG_SELECTED: Indicating that the node is selected.
  - SCCDA_TREENODEFLAG_FOCUS: Indicating that the node has focus.
  - SCCDA_TREENODEFLAG_ENCRYPT: Indicating that the node is encrypted and can not be decrypted.
  - SCCDA_TREENODEFLAG_ARCKNOWNENCRYPT: indicating that the node is encrypted with an unknown encryption and can not be decrypted.
  - SCCDA_TREENODEFLAG_BUFFEROVERFLOW: the name of the node was too long for the szName field.
- dwCharSet: Returns the SO_* (charsets.h) character set of the characters in szName. The output character set is either the default native environment character set or Unicode if the SCCOPT_SYSTEMFLAGS option is set to SCCVW_SYSTEM_UNICODE.

6.13 DAOpenTreeRecord

This function is called to open a record within an archive file and make it accessible by one or more of the data access technologies.

Search Export Only: Search Export's default behavior is to automatically open and process the contents of an archive. Use DAOpenTreeRecord and SCCOPT_XML_SEARCHML_FLAGS to change the default behavior if discrete processing of each document in an archive is desired.

```
DAERR DAOpenTreeRecord(
      VTHDOC      hDoc,
      VTLPHDOC    lphDoc,
      VTDWORD     dwRecord);
```

lphDoc is *not* a file handle.

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.).
- lphDoc: Pointer to a handle that is filled with a value uniquely identifying the document to data access. The developer uses this handle in subsequent calls to data access to identify this particular document.
- dwRecord: The record in the archive file to be opened.

Return Value

- DAERR_OK: Returned if DAOpenTreeRecord was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.14 DASaveTreeRecord

This function is called to extract a record in an archive file to disk.

```
DAERR DASaveTreeRecord(
      VTHDOC      hDoc,
      VTDWORD     dwRecord,
      VTDWORD     dwSpecType,
      VTLPVOID    pSpec,
      VTDWORD     dwFlags);
```

Parameters

- hDoc: Handle that uniquely identifies the document to data access. This is not an operating system file handle.
- dwRecord: The record in the archive file to be extracted.
- dwSpecType: Describes the contents of pSpec. Together, dwSpecType and pSpec describe the location of the source file to which the file will be extracted. Must be one of the following values:
  - IOTYPE_ANSIPATH: Windows only. pSpec points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) filename conventions.
  - IOTYPE_REDIRECT: Specifies that redirected I/O will be used to save the file.
  - IOTYPE_UNICODEPATH: Windows only. pSpec points to a NULL-terminated full path name using the Unicode character set and NTFS (Win32 and Win64) file name conventions.
  - IOTYPE_UNIXPATH: X Windows on UNIX platforms only. pSpec points to a NULL-terminated full path name using the system default character set and UNIX path conventions. Unicode paths can be accessed on UNIX platforms by using a UTF-8 encoded path with IOTYPE_UNIXPATH.
- pSpec: File location specification. See the descriptions for individual dwSpecType values.
- dwFlags: Currently not used. Should be set to 0.

Return Values

- DAERR_OK: Returned if the save was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_UNSUPPORTEDCOMP: Unsupported Compression Encountered.
- DAERR_PROTECTEDFILE: The file is encrypted.
- DAERR_BADPARAM: The request option is invalid. The record is possibly a directory.

Currently, only extracting a single file is supported. There is a known limitation where files in a Microsoft Binder file cannot be extracted.

6.15 DACloseTreeRecord

This function is called to close an open record file handle.

Search Export Only: Search Export's default behavior is to automatically open and process the contents of an archive. Use DACloseTreeRecord and SCCOPT_XML_SEARCHML_FLAGS to change the default behavior if discrete processing of each document in an archive is desired.

```
DAERR DACloseTreeRecord(
```

```
    VTHDOC      hDoc);
```

Parameters

- hDoc: Identifier of open record document.

Return Value

- DAERR_OK: Returned if DACloseTreeRecord was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.16 DASetStatCallback

This function sets up a callback that the technology periodically calls to verify the file is still being processed. The customer can use this with a monitoring process to help identify files that may be hung. Because this function is called more frequently than other callbacks, it is implemented as a separate function.

Use of the Status Callback Function

An application's status callback function will be called periodically by Oracle Outside In to provide a status message. Currently, the only status message defined is OIT_STATUS_WORKING, which provides a "sign of life" that can be used during unusually long processing operations to verify that Oracle Outside In has not stopped working. If the application decides that it would not like to continue processing the current document, it may use the return value from this function to tell Oracle Outside In to abort.

The status callback function has two return values defined:

- OIT_STATUS_CONTINUE: Tells Oracle Outside In to continue processing the current document.
- OIT_STATUS_ABORT: Tells Oracle Outside In to stop processing the current document.

The following is an example of a minimal status callback function.

```
VTDWORD MyStatusCallback( VTHANDLE hUnique, VTDWORD dwID, VTSYSVAL
pCallbackData, VTSYSVAL pAppData)
{
    if(dwID == OIT_STATUS_WORKING)
    {
        if( checkNeedToAbort( pAppData ) )
            return (OIT_STATUS_ABORT);
    }

    return (OIT_STATUS_CONTINUE);
}
```

Prototype

```
DAERR DASetStatCallback(DASTATCALLBACKFN pCallback,
   VTHANDLE hUnique,
   VTLPVOID pAppData)
```

Parameters

- pCallback: Pointer to the callback function.
- hUnique: Handle that may either be an hDoc or an hExport.
- pAppData: User-defined data. Oracle Outside In never uses this value other than to provide it to the callback function.

The callback function should be of type DASTATCALLBACKFN. This function has the following signature:

```
(VTHANDLE hUnique, VTDWORD dwID, VTSYSVAL pCallbackData, VTSYSVAL pAppData)
```

- hUnique: Handle that may either be an hDoc or an hExport
- dwID: Handle that indicates the callback status.
  - OIT_STATUS_WORKING
  - OIT_STATUS_CONTINUE
  - OIT_STATUS_CANCEL
  - OIT_STATUS_ABORT
- pCallbackData: Currently always NULL
- pAppData: User-defined data provided to DASetStatCallback

Return Values

- DAERR_OK: If successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

6.17 DASetFileAccessCallback

This function sets up a callback that the technology will call into to request information required to open an input file. This information may be the password of the file or a support file location.

Use of the File Access Callback

When the technology encounters a file that requires additional information to access its contents, the application's callback function will be called for this information. Currently, only two different forms of information will be requested: the password of a document, or the file used by Lotus Notes to authenticate the user information.

The status callback function has two return values defined:

- SCCERR_OK: Tells Oracle Outside In that the requested information is provided.
- SCCERR_CANCEL: Tells Oracle Outside In that the requested information is not available.

This function will be repeatedly called if the information provided is not valid (such as the wrong password). It is the responsibility of the application to provide the correct information or return SCCERR_CANCEL.

Prototype

```
DAERR DASetFileAccessCallback (DAFILEACCESSCALLBACKFN pCallback);
```

Parameters

- pCallback: Pointer to the callback function.

Return Values

- DAERR_OK: If successful. Otherwise, one of the other DAERR_ values defined in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

The callback function should be of type DAFILEACCESSCALLBACKFN. This function has the following signature:

```
typedef VTDWORD (* DAFILEACCESSCALLBACKFN)(VTDWORD dwID, VTSYSVAL pRequestData,
VTSYSVAL pReturnData, VTDWORD dwReturnDataSize);
```

- dwID: ID of information requested:
  - OIT_FILEACCESS_PASSWORD: Requesting the password of the file
  - OIT_FILEACCESS_NOTESID: Requesting the Notes ID file location
- pRequestData: Information about the file.

```
typedef struct {
      VTDWORD   dwSize;           /* size of this structure */
      VTWORD    wFIId;            /* FI id of reference file */
      VTDWORD   dwSpecType;       /* file spec type */
      VTVOID   *pSpec;            /* pointer to a file spec */
      VTDWORD   dwRootSpecType;   /* root file spec type */
      VTVOID   *pRootSpec;        /* pointer to the root file spec */
      VTDWORD   dwAttemptNumber;  /* The number of times the callback has */
                                  /* already been called for the currently */
                                  /* requested item of information */
} IOREQUESTDATA, * PIOREQUESTDATA;
```

- pReturnData: Pointer to the buffer to hold the requested information – for OIT_FILEACCESS_PASSWORD and OIT_FILEACCESS_NOTESID, the buffer is an array of WORD characters.
- dwReturnDataSize: Size of the return buffer.

Note: Not all formats that use passwords are supported. Only Microsoft Office binary (97-2003) nd Microsoft Office 2007, Lotus NSF, PDF (with RC4 encryption), nd Zip (with AES 128 & 256 bit, ZipCrypto) are currently supported.

6.18 DAOpenSubdocumentById

This function allows the caller to open subdocuments (for example, attachments within an email message file) for exporting as separate files.

```
DAERR DAOpenSubdocumentById( VTHDOC hDoc, VTLPHDOC phDoc, VTDWORD dwSubdocId,
VTDWORD dwIdType )
```

- hDoc: Handle to a currently open document, obtained via DAOpenDocument .
- phDoc: Points to a VTHDOC value to be set by this function upon success.
- dwSubdocId: An identifier of the subdocument to be opened.
- dwIdType: A value that describes which type of identifier is specified by dwSubdocId Must be one of the following:
  - DASUBDOCID_ATTACHMENTINDEX: dwSubdocId is the zero-based index of the an attachment from an email message file. The first attachment presented by Outside has the index value 0, the second has the index value 1, etc.
  - DASUBDOCID_XX: dwSubdocId is a value provided by the output of XML Export. It is the value of the object_id attribute of a locator element.

Return Values

Returns DAERR_OK on success, or an appropriate error value if an error occurred.

6.19 DAAddOptionItem

This function adds a new item to the end of the list of items associated with a mult-value option. The value of the item id is determined internally by the Outside In code, and will not change for the lifetime of the option item.

If the caller is not interested in the value of the itemId, calling DASetOption multiple times is equivalent to calling DAAddOptionItem with the pItemId set to NULL. (This only applies to options whose values may be set through DAAddOptionItem).

```
SCCERR DAAddOptionItem (VTHDOC hDoc, DWORD dwOptionId, VTLPVOID pValue,
         VTDWORD dwSize, VTLPDWORD pItemId )
```

- hDoc: Handle to current document, as described in documentation of DASetOption.
- dwOptionId: The option id of an option that requires a list of values.
- pValue: Pointer to the value of the option item being added.
- dwSize: Size of the data pointed to by pValue.
- pItemId: Points to a DWORD that will receive the value of an internal identifier of the item. This identifer may be used in subsequent calls to DARemoveOptionItem. This parameter may be NULL if the caller is not interested in the id.

6.20 DAGetOptionItem

The item id value provided by this function is the same one provided by DAAddOptionItem when the item was first added. This function should not be called simultaneously for the same hDoc from two different threads.

```
SCCERR DAGetOptionItem( VTHDOC hDoc, DWORD dwOptionId, DWORD dwWhichItem,
         VTLPVOID pValue, VTLPDWORD pSize, VTLPDWORD pItemId )
```

- hDoc: Handle to current document, as described in documentation of DAGetOption.
- dwOptionId: The option id of an option whose item values are being requested.
- dwWhichItem: Must be one of the following:
  - SCCOPT_FIRSTITEM - which retrieves the first item in the specified list.
  - SCCOPT_NEXTITEM - which retrieves the next item in the specified list.
  - SCCOPT_ITEMSIZE - which does not retrieve an item, but sets *pSize to the necessary buffer size for an item of the specified list. (If the list items vary in size, it will indicate the size of the largest item.)
  - .SCCOPT_LISTSIZE - which does not retrieve an item, but sets *pSize to the count of all items in the specified list
- pValue: Pointer to a buffer that will receive the requested item's value.
- pSize: Size of the buffer pointed to by pValue. If this size isn't sufficient to receive the requested data, the function will set the value pointed to by pSize to the size required to hold the item's data, return SCCERR_INSUFFICIENTBUFFER.
- pItemId: Points to a DWORD that will receive the value of an internal identifier of the item. This identifer may be used in subsequent calls to DARemoveOptionItem. This parameter may be NULL if the caller is not interested in the id.

6.21 DARemoveOptionItem

This function should not be called simultaneously for the same hDoc from two different threads.

```
SCCERR DARemoveOptionItem( VTHDOC hDoc, DWORD dwOptionsId, DWORD dwItemId )
```

- hDoc: Handle to current document, as described in documentation of DASetOption.
- dwOptionsId: The option id of an option that requires a list of values.
- dwItemId: An identifier to an option id, provided by either DAAddOptionItem or DAGetOptionItem, or SCCOPT_ALLITEMS. If SCCOPT_ALLITEMS is specified, all of the items associated with the specified option id will be removed.

Note: This function should not be called simultaneously for the same hDoc from two different threads.

7

Export Functions

This chapter outlines the basic functions used to initiate the conversion of documents using the product API.

This chapter includes the following sections:

7.1 General Functions

The following functions are general functions used in most export products.

This section includes the following functions:

7.1.1 EXOpenExport

This function is used to initiate the export process for a file that has been opened by DAOpenDocument. If EXOpenExport succeeds, EXCloseExport must be called regardless of any other API calls.

> **Note:** SCCOPT_GRAPHIC_TYPE = FI_NONE must be set (via DASetOption) before the call to EXOpenExport. Otherwise, the SCCUT_FILTEROPTIMIZEDFORTEXT speed enhancement for the PDF filter is not set. This will result in slower exports of PDFs when graphic output is not required.

Prototype

```
SCCERR EXOpenExport(
   VTHDOC       hDoc,
   VTDWORD      dwOutputId,
   VTDWORD      dwSpecType,
   VTLPVOID     pSpec,
   VTDWORD      dwFlags,
   VTSYSPARAM   dwReserved,
```

```
VTLPVOID     pCallbackFunc,
VTSYSPARAM   dwCallbackData,
VTLPHEXPORT  phExport);
```

Parameters

- hDoc: A handle that identifies the source file, created by DAOpenDocument. Knowledge of this should only affect OEMs under the most unusual of circumstances.
- dwOutputId: File ID of the desired format of the output file. This value must be set to either FI_PDF (for generic PDF 1.4), FI_PDFA (for PDF/A-1a compliance), or FI_PDFA_2 (for PDF/A-2a compliance).

  Note: For FI_PDFA exports, raster images with transparency will **not** be produced as transparent due to the explicit exclusion of transparency in the ISO PDF/A-1a specification document.

- dwSpecType: Describes the contents of pSpec. Together, dwSpecType and pSpec describe the location of the initial output file. Must be one of the following values:
  - IOTYPE_ANSIPATH: Windows only. pSpec points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) file name conventions.
  - IOTYPE_UNICODEPATH: Windows only. pSpec points to a NULL-terminated full path name using the Unicode character set and NTFS file name conventions.

    Note: If you are using IOTYPE_UNICODEPATH as a file spec type, if the calling application is providing an export callback function, you should set the option SCCOPT_EX_UNICODECALLBACKSTR to TRUE. Refer to the documentation on callbacks such as EX_CALLBACK_ID_CREATENEWFILE and the EXURLFILEIOCALLBACKDATAW structure for details

  - IOTYPE_UNIXPATH: UNIX platforms only. pSpec points to a NULL-terminated full path name using the system default character set and UNIX path conventions. Unicode paths can be accessed on UNIX platforms by using a UTF-8 encoded path with IOTYPE_UNIXPATH.
  - IOTYPE_REDIRECT: All platforms. pSpec may be NULL, and all file information specified in the callback routine. This allows the developer to redirect the IO routines used to write the files. For more information, see Chapter 8, "Redirected IO."
- pSpec: Initial output file location specification. This is either a pointer to a buffer or NULL.

  If the pointer is not NULL, the file referred to by the pSpec is assumed to be already open and the buffer's contents are based on the value of the dwSpecType parameter. See the descriptions for individual dwSpecType values in the preceding list.

  Passing NULL indicates the developer will use the EX_CALLBACK_ID_CREATENEWFILE callback to specify the initial output file instead of specifying it

here. When this parameter is NULL, the developer must handle the EX_CALLBACK_ID_CREATENEWFILE callback or EXOpenExport returns an error.

- dwFlags: Must be set by developer to 0.
- dwReserved: Reserved. Must be set by developer to 0.
- pCallbackFunc: Pointer to a function of the type EXCALLBACKPROC. This function is used to give the developer control of certain aspects of the export process as they occur. For more details, see the definition for EXCALLBACKPROC in Section 7.1.2, "EXCALLBACKPROC." This parameter may be set to NULL if the developer does not wish to handle callbacks.
- dwCallbackData: This parameter is passed transparently to the function specified by pCallbackFunc. The developer may use this value for any purpose, including passing context information into the callback function.
- phExport: Pointer to a handle that receives a value uniquely identifying the document to the product routines. If the function fails, this value is set to VTHDOC_INVALID. phExport is *not* a file handle.

Return Values

- SCCERR_OK: If the open was successful. Otherwise, one of the other SCCERR_ values in sccerr.h is returned.

7.1.2 EXCALLBACKPROC

Type definition for the developer's callback function.

Prototype

```
DAERR (DA_ENTRYMODPTR EXCALLBACKPROC)(
   VTHEXPORT    hExport,
   VTSYSPARAM   dwCallbackData,
   VTDWORD      dwCommandOrInfoId,
   VTLPVOID     pCommandOrInfoData);
```

Parameters

- hExport: Export handle for the document. Must be a handle returned by the EXOpenExport function.
- dwCallbackData: This value is passed to EXOpenExport in the dwCallbackData parameter.
- dwCommandOrInfoId: Indicates the type of callback. For information about supported callbacks, see Chapter 9, "Callbacks."
- pCommandOrInfoData: Data associated with dwCommandOrInfoId. For information about supported callbacks, see Chapter 9, "Callbacks."

Return Values

- SCCERR_OK: Command was handled by the callback function.
- SCCERR_BADPARAM: One of the function parameters was invalid.
- SCCERR_NOTHANDLED: Callback function did not handle the command. This return value must be the default for all values of dwCommandOrInfoId the developer does not handle.

7.1.3 EXCloseExport

This function is called to terminate the export process for a file.

Prototype

```
SCCERR EXCloseExport(
   VTHEXPORT   hExport);
```

Parameters

- hExport: Export handle for the document. Must be a handle returned by the EXOpenExport function.

Return Values

- SCCERR_OK: Returned if the close was successful. Otherwise, one of the other SCCERR_ values in sccerr.h is returned.

7.1.4 EXRunExport

This function is called to run the export process.

Prototype

```
SCCERR EXRunExport(
   VTHEXPORT   hExport);
```

Parameters

- hExport: Export handle for the document. Must be a handle returned by the EXOpenExport function.

Return Values

- SCCERR_OK: Returned if the export was successful. Otherwise, one of the other SCCERR_ values in sccerr.h is returned.

7.1.5 EXExportStatus

This function is used to determine if there were conversion problems during an export. It returns a structure that describes areas of a conversion that may not have high fidelity with the original document.

Prototype

```
SCCERR EXExportStatus(VTHEXPORT hExport, VTDWORD dwStatusType, VTLPVOID pStatus)
```

Parameters

- hExport: Export handle for the document.
- dwStatusType: Specifies which status information should be filled in pStatus.
  - EXSTATUS_SUBDOC – fills in the EXSUBDOCSTATUS structure (only implemented in Search Export and XML Export)
  - EXSTATUS_INFORMATION - fills in the EXSTATUSINFORMATION structure.
- pStatus: Either a pointer to a EXSUBDOCSTATUS or EXSTATUSINFORMATION data structure depending on the value of dwStatusType.

Return Values

SCCERR_OK: Returned if there were no problems. Otherwise, one of the other SCCERR_ values in sccerr.h is returned.

EXSUBDOCSTATUS Structure

The EXSUBDOCSTATUS structure is defined as follows:

```
typedef struct EXSUBDOCSTATUStag
{
VTDWORD dwSize;      /* size of this structure */
VTDWORD dwSucceeded; /* number of sub documents that were converted */
VTDWORD dwFailed;    /* number of sub documents that were not converted */
} EXSUBDOCSTATUS;
```

EXSTATUSINFORMATION Structure

The EXSTATUSINFORMATION structure is defined as follows:

```
typedef struct EXSTATUSINFORMATIONtag
{
VTDWORD dwVersion;              /* version of this structure, currently
EXSTATUSVERSION1      */
VTBOOL bMissingMap;             /* a PDF text run was missing the toUnicode table
*/
VTBOOL bVerticalText;           /* a vertical text run was present */
VTBOOL bTextEffects;            /* unsupported text effects applied (i.e.Word
Art)*/
VTBOOL bUnsupportedCompression; /* a graphic had an unsupported compression */
VTBOOL bUnsupportedColorSpace;  /* a graphic had an unsupported color space */
VTBOOL bForms;                  /* a sub documents had forms */
VTBOOL bRightToLeftTables;      /* a table had right to left columns */
VTBOOL bEquations;              /* a file had equations*/
VTBOOL bAliasedFont;            /* A font was missing, but a font alias was used
*/
VTBOOL bMissingFont;            /* The desired font wasn't present on the system
*/
VTBOOL bSubDocFailed;           /* a sub document was not converted */
} EXSTATUSINFORMATION;
```

```
#define EXSTATUSVERSION1     0X0001
```

> **Note:** When processing the main document, Search Export, HTML Export, and XML Export never use fonts, so bAliasedFont and bMissingFont will never report TRUE; however, when doing graphics conversions XML Export and HTML Export may use fonts, so bAliasedFont and bMissingFont may report TRUE.

7.1.6 EXAddKeyValue Functions

These functions specify key/value pairs for use in the exported document.

```
SCCERR EXAddKeyValueString( HEXPORT hExport, VTLPCSTR szKeyName, VTLPCSTR szValue
)
SCCERR EXAddKeyValueInt( HEXPORT hExport, VTLPCSTR szKeyName, VTDWORD dwValue)
SCCERR EXAddKeyValueFloat( HEXPORT hExport, VTLPCSTR szKeyName, VTFLOAT fValue )
```

These functions are being added to Outside In 8.5.0 for the use of the Web View Export product. They may in the future be used in other SDKs. A given key name cannot have

multiple values. Calling this function multiple times with the same value for szKeyName will replace each previous value with the newest one.

Parameters

- hExport: Export handle for the document. Must be a handle returned by the EXOpenExport function.
- szKeyName: A name for the data (as a null-terminated UTF8-encoded string).
- szValue / dwValue / fValue: The value side of the key-value pair - either a null-terminated UTF8-encoded string or a VTDWORD or a VTFLOAT.

Return Values

- SCCERR_OK or an appropriate error value.

Use in Web View Export Output

The data specified by this method will be made available to scripts that have access to the Web View Export Javascript API. For a script operating within a web view, the data specified will be accessible through an object returned from the API function OIT.document.externalData().

For example, if at export time the export SDK is called like this:

```
EXAddKeyValueInt( hExport, (VTLPCSTR)"UserId", 42 );
EXAddKeyValueString( hExport, (VTLPCSTR)"UserName", (VTLPCSTR)"Bob" );
EXAddKeyValueFloat( hExport, (VTLPCSTR)"Pie", 3.14159 );
```

Then from a script that is loaded into the Web View Export output, the following is possible:

```
var myData = OIT.document.externalData();

alert( myData.UserId ); // displays "42"
alert( myData["UserName"] ); // displays "Bob"
alert( myData.Pie ); // displays "3.14159"
```

7.2 Annotation Functions

Annotations are a way to highlight, insert, or delete text in product output, without modifying the original document.

This section covers the following annotation functions:

Examples of ways annotations can be used by developers include:

- highlighting search hits
- inserting notes to comment on text in the original document

- deleting sensitive information not intended for viewing

Other Oracle Outside In products are required to ascertain the proper character positions where the developer wishes to make annotations. Currently, only Content Access and the SearchML output format (available in Search Export) can be used to get these positions. Although the Content Access module is included with the product, license to use the Content Access API is not automatically granted with the purchase of the Export software.

A separate license for Content Access or Search Export is required to enable use of any of the annotation features that are supported by Web View Export. Contact your sales representative for more information.

The following notes should be considered when using annotations:

- Processing annotations slows down the conversion process to some extent.
- While other products in the Oracle Outside In family support annotations, not all products support all types of annotations.
- The ACC acronym (Actual Character Count) is used in the following function descriptions. ACCs represent the location of text in the source document data stream. They represent a marker just before the location of text, and this marker is zero-based.

  startACC parameters should be set to an ACC value that represents the position just prior to the first character and endACC parameters should be set to an ACC value that represents the position just past the last character in the range. For this reason, users should make sure endACC values are 1 greater than the ACC of the last character in the desired range of annotation.
- Calling EXCloseExport causes all annotations set so far to be cleared.

7.2.1 EXHiliteText

This function allows the developer to change foreground and background colors of a range of characters from the input document.

The colors set by this option can be overridden by the equivalent settings in the ExInsertText function.

Prototype

```
DAERR EXHiliteText(
VTHEXPORT           hExport,
PEXANNOHILITETEXT   pHiliteText);
```

Parameters

- hExport: Export handle for the document. Must be the handle returned by the EXOpenExport() function.
- pHiliteText: Pointer to a structure containing the information on what to highlight and how to highlight it.

Structure

A C data structure defined in sccex.h as follows:

```
typedef struct EXANNOHILITETEXTtag
{
   VTDWORD         dwSize;
   VTDWORD         dwStartACC;
```

```
   VTDWORD         dwEndACC;     /* Last char to highlight +1 */
   VTDWORD         dwOptions;
   SCCVWCOLORREF   sForeground;
   SCCVWCOLORREF   sBackground;
   VTWORD          wCharAttr;
   VTWORD          wCharAttrMask;
} EXANNOHILITETEXT;
```

- dwSize: Must be set by the developer to sizeof(EXANNOHILITETEXT).
- dwStartACC: The ACC of the first character to be highlighted.
- dwEndACC: ACC of the last character to be highlighted +1. Ranges for annotations have their end point set one past the ACC of the last character in the range. For example, to highlight a single character at ACC position 5, dwStartACC would be set to 5, and dwEndACC would be set to 5+1=6.
- dwOptions: Flags that provide highlight options. The default is all flags set to off. The valid flags are:
  - SCCVW_USEFOREGROUND: Indicates that sForeground defines the foreground text color to apply to highlights.
  - SCCVW_USEBACKGROUND: Indicates that sBackground defines the background text color to apply to highlights.
  - SCCVW_USECHARATTR: Indicates that wCharAttr defines the character attributes to apply to highlights.
  - sForeground: Defines the foreground text color to be used if the SCCVW_USEFOREGROUND flag is set in dwOptions. Set this value with the SCCANNORGB(red, green, blue) macro. The red, green and blue values are percentages of the color from 0-255 (with 255 being 100%). There is no default value for this parameter -- if it is set, the color must be specified.
  - sBackground: Defines the background text color to be used if the SCCVW_USEBACKGROUND flag is set in dwOptions. Set this value with the SCCANNORGB(red, green, blue) macro. The red, green and blue values are percentages of the color from 0-255 (with 255 being 100%). There is no default value for this parameter. If it is set, the color must be specified.
  - wCharAttr: Defines the character attributes to use if SCCVW_USECHARATTR is set in dwOptions. Only bits with the corresponding bits set in wCharAttrMask are affected. To turn off all character attributes, set this to SCCVW_CHARATTR_NORMAL (the default) and set wCharAttrMask to -1. Otherwise, set this to any of the following character attributes OR-ed together:
    * SCCVW_CHARATTR_UNDERLINE
    * SCCVW_CHARATTR_ITALIC
    * SCCVW_CHARATTR_BOLD
    * SCCVW_CHARATTR_STRIKEOUT
    * SCCVW_CHARATTR_SMALLCAPS: Not supported in **Web View Export**.
    * SCCVW_CHARATTR_OUTLINE: Not currently supported.
    * SCCVW_CHARATTR_SHADOW: Not currently supported.
    * SCCVW_CHARATTR_CAPS: Not currently supported.
    * SCCVW_CHARATTR_SUBSCRIPT

* SCCVW_CHARATTR_SUPERSCRIPT
* SCCVW_CHARATTR_DUNDERLINE
* SCCVW_CHARATTR_WORDUNDERLINE
* SCCVW_CHARATTR_DOTUNDERLINE: Currently supported as single underline.

- wCharAttrMask: Defines which character attributes to change based on the settings of the bits in wCharAttr. Uses the same bit flags defined above for wCharAttr. Only attributes whose flag is set in this mask are modified to match the state specified by wCharAttr. This mask provides a way to distinguish between bits being set in wCharAttr because the developer wants to force a change to the character attributes and bits in wCharAttr that the developer would rather set to "inherit from the source document." The following are real-world examples of these interactions (all examples assume that SCCVW_ USECHARATTR is set in dwOptions):
  - Example 1: wCharAttr is set to SCCVW_CHARATTR_BOLD and wCharAttrMask is set to SCCVW_CHARATTR_BOLD. This results in bold being forced on in the annotation.
  - Example 2: wCharAttr is set to SCCVW_CHARATTR_BOLD and wCharAttrMask is set to 0. This results in bold being left the way it was in the source document in the annotation.
  - Example 3: wCharAttr is set to 0 and wCharAttrMask is set to SCCVW_ CHARATTR_BOLD. This results in bold being forced off in the annotation.

    The default value for this is 0, meaning that all the flags in wCharAttr are ignored.

Return Values

- DAERR_OK: Returned if the annotation was successfully added. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

7.2.2 EXHiliteTextEx

```
DAERR EXHiliteTextEx( VTHEXPORT hExport, PEXANNOHILITETEXT pHilite,
        VTLPDWORD pHiliteId )
```

This function is the same as EXHiliteText, but allows for a highlight id to be obtained, so that highlight properties or a text comment can be attached to this highlight. (This function was called EXHiliteAnchorText in earlier specifications)

- hExport: Export handle
- pHilite: Specifies details of the highlight formatting. (Note that not all of these attributes may be presented in Calvin output.')
- pHiliteId: An identifier generated by Export. It may be used as a parameter to EXAddComment to associate the comment with this highlight.

7.2.3 EXHiliteArea

```
DAERR EXHiliteArea( VTHEXPORT hExport, PEXANNOHILITEAREA pHilite,
        VTLPDWORD pHiliteId)
```

This function applies an area highlight to the export. The highlight is defined through the following data structure.

- hExport: Export handle
- pHilite: Specifies details of the highlight position and formatting.
- pHiliteId: An identifier generated by Export. It may be used as a parameter to EXAddComment to associate the comment with this highlight.

7.2.3.1 EXANNOHILITEAREA

```
typedef struct EXANNOHILITEAREA
{
  VTDWORD    dwSize;
  VTDWORD    dwSection;         // zero based number of (sheet/image/slide)
  VTDWORD    dwTop;             // Top coordinate or row
  VTDWORD    dwLeft;            // Leftmost coordinate or column
  VTDWORD    dwWidth;           // Width of area in coordinates or columns
  VTDWORD    dwHeight;          // Height of area in coordinates or columns
  VTDWORD    dwUnits;           // SCCANNO_TWIPS, SCCANNO_PIXELS or SCCANNO_CELLS
  VTDWORD    dwUser;            // User data

  SCCVWCOLORREF fillColor;      // Fill color
  VTFLOAT    fOpacity;          // 0-1.0; 0==invisible; applies to fill color *

  SCCVWCOLORREF borderTopColor;      // Border Colors and thickness in twips *
  VTDWORD       borderTopThickness;
  SCCVWCOLORREF borderLeftColor;
  VTDWORD       borderLeftThickness;
  SCCVWCOLORREF borderBottomColor;
  VTDWORD       borderBottomThickness;
  SCCVWCOLORREF borderRightColor;
  VTDWORD       borderRightThickness;

  VTDWORD       dwBorderStyle;      // Currently not supported
                                    // (would be SCCVW_BORDER_DOT, SCCVW_BORDER_
DASH, or SCCVWBORDER_SOLID)

} EXANNOHILITEAREA;
```

7.2.4 EXAddStampAnnotation

```
DAERR EXAddStampAnnotation( VTHEXPORT hExport, PEXANNOSTAMP pStamp
         VTLPDWORD pHiliteId)
```

This function applies an image stamp to the export output. The stamp is defined through the following data structure.

- hExport: Export handle
- pStamp: Specifies details of the image and its position
- pHiliteId: An identifier generated by Export. It may be used as a parameter to EXAddComment to associate the comment with this highlight.

7.2.4.1 EXANNOSTAMP

```
typedef struct EXANNOSTAMP *
{
  VTDWORD    dwSize;
  VTDWORD    dwUser;            // User data
```

```
  VTDWORD    dwSection;         // zero based number of (page/sheet/image/slide)
  VTDWORD    dwTop;             // Top coordinate or row
  VTDWORD    dwLeft;            // Leftmost coordinate or column
  VTDWORD    dwWidth;           // Width of area in coordinates or columns
  VTDWORD    dwHeight;          // Height of area in coordinates or columns
  VTDWORD    dwUnits;           // EX_ANNO_TWIPS, EX_ANNO_PIXELS or EX_ANNO_CELLS
  VTFLOAT    fOpacity;          // 0-1.0; 0==invisible

  DWORD      dwSizeMode;        // EX_ANNO_SIZE_FIT, EX_ANNO_SIZE_STRETCH,
                                // EX_ANNO_SIZE_FROM_SOURCE;
  VTLPCSTR   szStampName;       // name of stamp image, specified via one of
                                // the SCCOPT_STAMP_IMAGE_* options

} EXANNOSTAMP;
```

Notes

For word processing files, the dwSection parameter indicates the zero-based index of the page of the output document to which the stamp should be applied. Note that the usual caveats apply to paginated output. Pagination is influenced by how text is wrapped on a page, which is in turn influenced by the fonts that are available to the Outside In rendering code on the system that is generating the output. Exporting the same document on two different systems with different fonts available to them may result in slightly different page boundaries.

About the dwSizeMode parameter:

- If this is set to EX_ANNO_SIZE_FIT, the source image will be scaled with its aspect ratio preserved, to fit inside the rectangle specified by dwTop, dwLeft, dwWidth, and dwHeight. The top, left corner of the scaled image will align with the top, left corner of the highlight rectangle.
- If this is set to EX_ANNO_SIZE_STRETCH, the source image will be stretched to fill the highlight rectangle.
- If this is set to EX_ANNO_SIZE_FROM_SOURCE, the source image will be rendered at its native size. The dwWidth and dwHeight fields will be ignored, and the top, left corner of the stamped image will be aligned with the top, left corner of the destination rectangle.

7.2.5 EXAddComment

```
DAERR EXAddComment( VTHEXPORT hExport, VTDWORD dwHighlightId, VTLPCWSTR pComment )
```

- hExport: Export handle
- dwHighlightId: Id value provided by EXHighlightTextEx, EXHighlightArea, or EXAddStampAnnotation
- pComment: Comment text, null-terminated Unicode UCS2 string

This function adds a comment associated with a previously applied highlight annotation. If a comment was previously defined for the highlight, it is replaced by the new comment.

7.2.6 EXAddHiliteProperty

```
DAERR EXAddHiliteProperty( HEXPORT hExport, DWORD dwAnchorId, VTLPCWSTR wsName,
VTLPCWSTR sValue )
```

Note: This method associates a property with a highlight. The property is specified as a Unicode (UCS2) name/value pair.

7.2.7 EXApplyHilites

```
DAERR EXApplyHilites(HEXPORT hExport, const VTBYTE * pHilites);
```

This function applies a set of highlights from a JSON-encoded text stream previously generated from the Calvin Javascript library. This function may be called multiple times with different sets of highlights.

- hExport: Export handle
- pHilites: Buffer containing a stream of highlight (and comment) information that was obtained via the Calvin Javascript API function OIT.highlights.serialize.

8

Redirected IO

Anywhere a file specification (dwSpecType and pSpec parameters) is passed to a function in the product, the developer may use Redirected IO to completely take over responsibility for the low level IO calls of that particular file. The source file and all output files can be redirected in this way.

Redirected IO allows the developer great flexibility in the storage of, and access to, converted documents. For example, documents may be stored on file systems not supported natively by the software, or in a unique directory tree structure determined by the type of file.

When using Web View Export, redirected IO can also be used in conjunction with callbacks (discussed in Chapter 9, "Callbacks").

This chapter includes the following sections:

8.1 Using Redirected IO

A developer can redirect the IO for an input or output file by providing a data structure that contains pointers to custom IO routines for reading and writing. This data structure is passed in place of a typical file specification. The developer must set the dwSpecType parameter of the DAOpenDocument call to IOTYPE_REDIRECT when the DAOpenDocument call is sent.

When dwSpecType is set this way, the pSpec element must contain a pointer to a developer-defined data structure that begins with a BASEIO structure (defined in baseIO.H). The BASEIO structure contains pointers to the basic IO functions for the IO system such as Read, Seek, Tell, etc. The developer must initialize these function pointers to their own functions that perform IO tasks. Beyond the BASEIO element, the developer may place any data he or she likes.

For instance, a developer's structure may be similar to the following:

```
typedef struct MYFILEtag
{
   BASEIO    sBaseIO;       /* must be the first element */
   VTDWORD   dwMyInfo1;
   VTDWORD   dwMyInfo2;
   .
   .
   .
} MYFILE;
```

Because the pSpec passed is essentially the "file handle" used by the software, the developer can redirect the IO on a file-by-file basis while still exporting "regular" disk-based files.

The BASEIO structure is defined as follows:

```
typedef struct BASEIOtag
{
    IOCLOSEPROC pClose;
    IOREADPROC pRead;
    IOWRITEPROC pWrite;
    IOSEEKPROC pSeek;
    IOTELLPROC pTell;
    IOGETINFOPROC pGetInfo;
    IOOPENPROC pOpen; /* pOpen *MUST* be set to NULL. */
#ifndef NLM
    IOSEEK64PROC pSeek64;
    IOTELL64PROC pTell64;
#endif
    VTVOID *aDummy[3];
} BASEIO, * PBASEIO;
```

The developer must implement the Close, Read, Write, Seek, Tell and GetInfo routines. The Open routine must be set to NULL. The first parameter to each of these routines is called hFile and is of the type HIOFILE. HIOFILE is simply the VTLPVOID to your data structure that was passed in the pSpec parameter of the DAOpenDocument call.

The sample source code for a simple implementation of Redirected IO is in the samples directory. This sample redirects the technology's IO through the fopen, fgetc, fseek, ftell and fclose run-time library routines.

Important: Redirected IO does not cache the whole file. Seeks can occur throughout the file during the course of conversion. If the developer is implementing redirected IO on a slow or sequential link, it is the developer's responsibility to cache the file locally.

8.2 Opening Files

The developer does not see a call to pOpen when using redirected IO. When IOTYPE_REDIRECT is used, the structure passed in pSpec is defined to represent a file that is already open. The software can immediately call the pRead, pSeek, pTell and pWrite functions.

Files specified as using redirected IO must be open by the time they are handed off to the software.

8.3 IOClose

Closes the file identified by hFile and cleans up all memory associated with the file.

If you dynamically allocate your own file structures (MYFILE in the preceding discussion) it is required that the memory allocated be freed inside the call to IOClose or sometime thereafter.

Prototype

```
IOERR IOClose(
   HIOFILE   hFile);
```

Parameters

- hFile: Identifies the file to be closed. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).

Return Values

- IOERR_OK: Close was successful.
- IOERR_UNKNOWN: Some error occurred on close.

8.4 IORead

Reads data from the current file position forward and resets the position to the byte after the last byte read.

Prototype

```
IOERR IORead(
   HIOFILE         hFile,
   VTBYTE       *  pData,
   VTDWORD         dwSize,
   VTDWORD      *  pCount);
```

Parameters

- hFile: Identifies the file to be read. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).
- pData: Points to the buffer into which the bytes should be read. Will be at least dwSize bytes big.
- dwSize: Number of bytes to read.
- pCount: Points to the number of bytes actually read by the function. This value is only valid if the return value is IOERR_OK.

Return Values

- IOERR_OK: Read was successful. pCount contains the number of bytes read and pData contains the bytes themselves.
- IOERR_EOF: Read failed because the file pointer was beyond the end of the file at the time of the read.
- IOERR_UNKNOWN: Read failed for some other reason.

8.5 IOWrite

Writes data from the current file position forward and resets the position to the byte after the last byte written.

Prototype

```
IOERR IOWrite(
   HIOFILE         hFile,
   VTBYTE       *  pData,
   VTDWORD         dwSize,
   VTDWORD       * pCount);
```

Parameters

- hFile: Identifies the file where the data is to be written. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).
- pData: Points to the buffer from which the bytes should be written. It must be at least dwSize bytes big. It is good practice to treat the data passed in by pData as "read only." This helps prevent unexpected behavior elsewhere in the system.
- dwSize: Number of bytes to write.
- pCount: Points to the number of bytes actually written by the function. This value is only valid if the return value is IOERR_OK.

Return Values

- IOERR_OK: Write was successful, pCount contains the number of bytes written.
- IOERR_UNKNOWN: Write failed for some reason.

8.6 IOSeek

Moves the current file position.

Prototype

```
IOERR IOSeek(
   HIOFILE   hFile,
   VTWORD    wFrom,
   VTLONG    lOffset);
```

Parameters

- hFile: Identifies the file to be read. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).
- wFrom: One of the following values:
  - IOSEEK_TOP: Move the file position lOffset bytes from the top (beginning) of the file.
  - IOSEEK_BOTTOM: Move the file position lOffset bytes from the bottom (end) of the file.
  - IOSEEK_CURRENT: Move the file position lOffset bytes from the current file position.
- lOffset: Number of bytes to move the file pointer. A positive value moves the file pointer forward in the file and a negative value moves it backward. If a requested seek value would move the file pointer before the beginning of the file, the file

pointer should remain unchanged and IOERR_UNKNOWN should be returned. Seeking past EOF is allowed. In that case IOERR_OK should be returned. IOTell would return the requested seek position and IORead should return IOERR_EOF and 0 bytes read.

Return Values

- IOERR_OK: Seek was successful.
- IOERR_UNKNOWN: Seek failed for some reason.

8.7 IOTell

Returns the current file position.

Prototype

```
IOERR IOTell(
   HIOFILE         hFile,
   VTDWORD       * pOffset);
```

Parameters

- hFile: Identifies the file to be read. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).
- pOffset: Points to the current file position returned by the function.

Return Values

- IOERR_OK: Tell was successful.
- IOERR_UNKNOWN: Tell failed for some reason.

8.8 IOGetInfo

Returns information about an open file.

Prototype

```
IOERR IOGetInfo(
   HIOFILE        hFile,
   VTDWORD        dwInfoId,
   VTVOID       * pInfo);
```

Parameters

- hFile: Identifies the file to be read. Should be cast into a pointer to your data structure (MYFILE in the previous discussion).
- dwInfoId: One of the following values:
  - IOGETINFO_FILENAME: pInfo points to a string that should be filled with the base file name (no path) of the open file (for example TEST.DOC). If you do not know the file name, return IOERR_UNKNOWN. Certain file types (such as DataEase) must know the original file name in order to open secondary files required to correctly view the original file. If you return IOERR_UNKNOWN, these file types do not convert. For more information, see Section 8.8.1, "IOGENSECONDARY and IOGENSECONDARYW Structures."

- IOGETINFO_PATHNAME: pInfo points to a string that should be filled with the fully qualified path name (including the file name) of the open file. For example, C:\MYDIR\TEST.DOC. If you do not know the path name, return IOERR_UNKNOWN.
- IOGETINFO_PATHTYPE: pInfo points to a DWORD that should be filled with the IOTYPE of the path returned by IOGETINFO_PATHNAME. For instance, if you return a DOS path name in the Unicode character set, you should return IOTYPE_UNICODEPATH. Even if redirected IO is in use, this should not be set to IOTYPE_REDIRECT. The value should reflect the style of path to be returned or any other values detailed in Section 7.1.1, "EXOpenExport."
- IOGETINFO_ISOLE2STORAGE: Must return IOERR_FALSE. pInfo is not used.
- IOGETINFO_GENSECONDARY: pInfo points to a structure of type IOGENSECONDARY. Some file types require supporting files to be opened. These supporting files may contain formatting information or extra data. When using HTML Export, templates may link to other templates, and the paths to those templates must be resolved. Correct handling of IOGETINFO_GENSECONDARY is critical to the operation of the Oracle Outside In technology. For a list of these file types, see Section 8.8.2, "File Types That Cause IOGETINFO_GENSECONDARY."

  Because the developer is in total control of the IO for the primary file, the technology does not know how to generate a path to these secondary files or even if the secondary files are accessible through the regular file system. The IOGETINFO_GENSECONDARY call gives the developer a chance to resolve this problem by generating a new IO specification for the secondary file in question. The developer gets just the base file name (often embedded in the original document or generated from the primary file's name) of the secondary file.

  The developer may either use one of the standard Oracle Outside In IO types or totally redirect the IO for the secondary file, as well. For more details, see Section 8.8.1, "IOGENSECONDARY and IOGENSECONDARYW Structures."
- IOGETINFO_SUBDOC_SPEC: This message should be handled only if the currently open file is an archive and a particular item within the archive is intended to be specified as the input file in a call to DAOpenDocument. In this case, pInfo points to a single-byte character string that should be filled with the subdocument specification of an item within the open file. For example, item.2 specifies item 2 within the archive file. When specifying a subdocument specification, return IOERR_OK. Any other return values cause the results of this message to be ignored.
- IOGETINFO_64BITIO: For redirected I/O that wishes to use 64-bit seek/tell functions, your IOGetInfo function must respond IOERR_TRUE to this dwInfoId. In addition, the pSeek64/pTell64 items in the baseio structure must be valid pointers to the proper function types.
- IOGETINFO_DPATHNAME: pInfo points to a structure of type DPATHNAME, which should be filled with the fully qualified path name (including the file name) of the open file, for example, C:\MYDIR\TEST.DOC. If you do not know the path name, return IOERR_UNKNOWN. The dwPathLen element contains the size of the buffer pointed to by the pPath element. If the buffer size is too small to contain the full path, modify dwPathLen to be the correct size of the buffer required to hold the path name

in its IOTYPE character width including the NULL terminator and return IOERR_INSUFFICIENTBUFFER.

The following is a C data structure defined in SCCIO.H:

```
typedef struct DPATHNAMEtag
{
    VTDWORD  dwPathLen;
    VTVOID  *pPath;
} DPATHNAME, * PDPATHNAME;
```

Parameters

dwPathLen: Will be set to the number of bytes in the buffer pointed to by pPath. If the size of the buffer is insufficient, reset this element to the number of bytes required and return IOERR_INSUFFICIENTBUFFER.

pPath: Points to the buffer to be filled with the path name.

– IOGETINFO_GENSECONDARYDP: pInfo points to a structure of type IOGENSECONDARYDP. The dwSpecLen element contains the size of the buffer pointed to by the pSpec element. If the buffer size is too small to contain the spec, modify dwSpecLen to be the correct size of the buffer required to hold the path in its IOTYPE character width including the NULL terminator and return IOERR_INSUFFICIENTBUFFER.

The following is a C data structure defined in SCCIO.H:

```
typedef struct IOGENSECONDARYDPtag
{
    VTDWORD         dwSize;
    VTVOID *        pFileName;
    VTDWORD         dwSpecType;
    VTVOID *        pSpec;
    VTDWORD         dwSpecLen;
    VTDWORD         dwOpenFlags;
} IOGENSECONDARYDP, * PIOGENSECONDARYDP;
```

Parameters

dwSize: Will be set to sizeof (IOGENSECONDARYDP)

pFileName: A pointer to a string representing the file name of the secondary file that the technology requires. It is usually a name stored in the primary file (such as MYSTYLE.STY for a Word for DOS file) or a name generated from the primary file name. The primary file for a DataEase database has a .dba extension. The secondary name is the same file name but with a .dbm extension.

dwSpecType: The developer must fill this with the IOSPEC for the secondary file.

pSpec: On entry, this pointer points to an array of bytes or may be NULL (see dwSpecLen below). If the dwSpecType is set a regular IOTYPE such as IOTYPE_ANSIPATH, the developer may fill this array with the path name or structure required for that IOTYPE. If the developer is redirecting access to the secondary file, then dwSpecType will be IOTYPE_REDIRECT and the developer should replace pSpec with a pointer to a developer-defined structure that begins with the BASEIO structure (see Section 8.1, "Using Redirected IO").

The file is supposed to be opened by the OEM's redirected IO code by the time they return the BASEIO struct. This is because the pOpen routine in the BASEIO struct is supposed to be NULL.

dwSpecLen: On entry, this is set to the size of the pSpec buffer. If the size of the buffer is insufficient, replace the value with the number of bytes required and return IOERR_INSUFFICIENTBUFFER.

dwOpenFlags: Set by the technology. A set of bit flags describing how the secondary file should be opened. Multiple flags may be used by bitwise OR-ing them together. The following flags are currently used:

- IOOPEN_READ: The secondary file should be opened for read.

- IOOPEN_WRITE: The secondary file should be opened for write. If the specified file already exists, its contents are erased when this flag is set.

- IOOPEN_CREATE: The secondary file should be created (if it does not already exist) and opened for write.

Any other value should return IOERR_BADINFOID.

- pInfo: The size of the pInfo buffer depends on the **dwInfoId** selected. For IOGETINFO_FILENAME and IOGETINFO_PATHNAME, the buffer is of size MAX_PATH characters (each character is either one byte or two, depending on PATHTYPE). The IOGETINFO_PATHTYPE buffer is the size of a VTDWORD.

Return Values

- IOERR_OK: GetInfo was successful.
- IOERR_TRUE: Affirmative response from a true or false GetInfo.
- IOERR_FALSE: Negative response from a true or false GetInfo.
- IOERR_BADINFOID: dwInfoId can not be handled by this file type.
- IOERR_INVALIDSPEC: The file spec is bad for this type.
- IOERR_UNKNOWN: GetInfo failed for some other reason.

8.8.1 IOGENSECONDARY and IOGENSECONDARYW Structures

These structures are passed to the developer through the IOGetInfo function. They allow the developer to tell the technology where a secondary file, needed by the conversion process, is located.

The SpecType of the original file determines which of these two structures is used. If the SpecType is IOTYPE_UNICODEPATH, IOGENSECONDARYW is used. pFileName points to a Unicode string terminated with a NULL WORD. For all other SpecTypes, IOGENSECONDARY is used and pFileName points to a string terminated with a NULL BYTE.

When using HTML Export, consider the situation where the software must access a secondary template file. In that case, the SpecType of the original template specified by the option SCCOPT_EX_TEMPLATE determines which of the two structures is used.

The following is a C data structure defined in SCCIO.H:

```
typedef struct
{
   VTDWORD     dwSize;
   VTLPBYTE    pFileName;
   VTDWORD     dwSpecType;
```

```
   VTLPVOID    pSpec;
   VTDWORD     dwOpenFlags
} IOGENSECONDARY, * PIOGENSECONDARY;

typedef struct
{
   VTDWORD     dwSize;
   VTLPWORD    pFileName;
   VTDWORD     dwSpecType;
   VTLPVOID    pSpec;
   VTDWORD     dwOpenFlags
} IOGENSECONDARYW, * PIOGENSECONDARYW;
```

Parameters

- dwSize: Will be set to sizeof (IOGENSECONDARY) or sizeof (IOGENSECONDARYW) (both of these values are the same).
- pFileName: A pointer to a string representing the file name of the secondary file that the technology requires. It is usually a name stored in the primary file (such as MYSTYLE.STY for a Word for DOS file) or a name generated from the primary file name. The primary file for a DataEase database has a .dba extension. The secondary name is the same file name but with a .dbm extension.
- dwSpecType: The developer must fill this with the IOSPEC for the secondary file.
- pSpec: On entry, this pointer points to an array of 1024 bytes. If the dwSpecType is set a regular IOTYPE such as IOTYPE_ANSIPATH, the developer may fill this array with the path name or structure required for that IOTYPE. If the developer is redirecting access to the secondary file, then dwSpecType will be IOTYPE_REDIRECT and the developer should replace pSpec with a pointer to a developer-defined structure that begins with the BASEIO structure (see Section 8.1, "Using Redirected IO").

  The file is supposed to be opened by the OEM's redirected IO code by the time they return the BASEIO struct. This is because the pOpen routine in the BASEIO struct is supposed to be NULL.
- dwOpenFlags: Set by the technology. A set of bit flags describing how the secondary file should be opened. Multiple flags may be used by bitwise OR-ing them together. The following flags are currently used:
  - IOOPEN_READ: The secondary file should be opened for read.
  - IOOPEN_WRITE: The secondary file should be opened for write. If the specified file already exists, its contents are erased when this flag is set.
  - IOOPEN_CREATE: The secondary file should be created (if it does not already exist) and opened for write.

8.8.2 File Types That Cause IOGETINFO_GENSECONDARY

The following file types cause IOGETINFO_GENSECONDARY:

- Microsoft Word for DOS Versions 4, 5 and 6: Used to open and read the style sheet file associated with the document. The filter degrades if the style sheet is not present.
- Harvard Graphics DOS 3.x: Used to open and read the individual slides within ScreenShow and palette files. Files with the extension .ch3 are individual graphics or slides that can be opened using no secondary files. Files with the extension .sy3 are ScreenShows that reference a list of .ch3 files via the secondary file mechanism.

There is also an optional palette file that can be referenced from a .ch3 file, but the filter degrades if the palette file is not present.

- R:Base: Used to open and read required schema file. The R:Base data files are named ????2.rbf but the data is useless without the schema file named ????1.rbf. There is also a ????3.rbf file associated with each database, but it is not used.
- Paradox 4.0 and Above: Used to open and read memo field data file. Paradox uses a separate file for all memo field data larger than 32 bytes.
- DataEase: Used to open and read the data file. DataEase databases include a .dba file that contains the schema (the file that the technology can identify as DataEase) and a .dbm file that contains the actual data.
- Templates (HTML Export): Any template that contains a `{## link}` will need to open the linked files. Additionally, when the root template is opened using redirected IO, each `{## copy}` macro in the template will result in a IOGETINFO_GENSECONDARY call, as well.

8.9 IOSEEK64PROC / IOTELL64PROC

These functions are for seek/tell using 64-bit offsets. These functions are not used by default. Rather, they are used if the IOGETINFO_64BITIO message returns IOERR_TRUE. This is so redirected I/O using strictly 32-bit I/O is unaffected.

8.9.1 IOSeek64

Moves the current file position.

Prototype

```
IOERR IOSeek64(
HIOFILE hFile,
VTWORD wFrom,
VTOFF_T offset);
```

Parameters

The parameter information is the same as for IOSeek(). However, the size of the VTOFF_T offset for IOSeek64() is 64-bit unlike the 32-bit offset in IOSeek().

8.9.2 IOTell64

Returns the current file position.

Prototype

```
IOERR IOTell64(
HIOFILE hFile,
VTOFF_T * pOffset);
```

Parameters

The parameter information is the same as for IOTell(). The only change is the use of a pointer to a 64-bit parameter for returning the offset.

9 Callbacks

Callbacks allow the developer to intervene at critical points in the export process. Read more about the callback procedure and the EXOpenExport function call in Section 7.1.1, "EXOpenExport." Each heading in this chapter is a possible value for the dwCommandOrInfoId parameter passed to the developer's callback.

The new SCCOPT_EX_CALLBACKS option allows developers to enable or disable some or all of these callbacks. See the Options documentation for details.

This section describes callbacks set in EXOpenExport. A second callback function, DASetStartCallback, can provide information about the progress of a file conversion. For more details, see Chapter 6, "Data Access Common Functions."

9.1 Callbacks Used In Web View Export

The following information applies to Web View Export.

This section includes the following callbacks:

- Section 9.1.1, "EX_CALLBACK_ID_CREATENEWFILE"
- Section 9.1.2, "EX_CALLBACK_ID_NEWFILEINFO"
- Section 9.1.3, "EX_CALLBACK_ID_PAGECOUNT"

9.1.1 EX_CALLBACK_ID_CREATENEWFILE

This callback is made any time a new output file needs to be generated. This gives the developer the chance to execute routines before each new file is created.

It allows the developer to override the standard naming for a file or to redirect entirely the IO calls for a file. This callback is made for all output files that are created. It does not include the already open initial file passed to EXOpen Export, unless of course redirected IO is in use with a pSpec of NULL.

If redirected IO is being used on output files, this callback must be implemented.

For this callback, the pCommandOrInfoData parameter points to a structure of type EXFILEIOCALLBACKDATA:

```
typedef struct EXFILEIOCALLBACKDATAtag
{
   HIOFILE    hParentFile;
   VTDWORD    dwParentOutputId;
   VTDWORD    dwAssociation;
   VTDWORD    dwOutputId;
   VTDWORD    dwFlags;
   VTDWORD    dwSpecType;
```

```
   VTLPVOID   pSpec;
   VTLPVOID   pExportData;
   VTLPVOID   pTemplateName;
} EXFILEIOCALLBACKDATA;
```

- hParentFile: Handle to the initial output file with which the new file is associated. The dwAssociation describes the relationship. This handle is not intended for use by the developer. Set by caller.
- dwParentOutputId: Set by caller. The type of the parent file. This value is FI_HTML5.
- dwAssociation: One of the following values:
  - CU_ROOT: For the initial output file.
  - CU_SIBLING: For new files that are not somehow owned by the parent file.
- dwOutputId: The type of the new file. This value is either FI_HTML5, FI_HTML_CSS, FI_JAVASCRIPT, or FI_PNG.
- dwFlags: Reserved
- dwSpecType: IO specification type. For details about IO specifications, see Section 6.3, "DAOpenDocument."

  This member in conjunction with pSpec allows the developer to choose any location for the new file or even redirect its IO calls entirely. For more details, see Chapter 8, "Redirected IO." When the developer receives this callback, the value of this element is undefined. Must be set by developer if this callback returns SCCERR_OK.
- pSpec: This field holds the IO specification of the output file to be created. pSpec points to a buffer that is 1024 bytes in size. If your application needs to set the specification of the output file, it may do so by either writing new data into this buffer, or by changing the value of pSpec to point to memory owned by your application. If pSpec is set to a new value, then your application must ensure that this memory stays valid for an appropriate length of time, at least until the next callback message is received, or EXRunExport returns.

  If the current export operation is using redirected IO, your application must create a redirected IO data structure for the new file and set pSpec to point to it. This pointer must stay valid until the structure's pClose function is called.

  If your application sets dwSpecType to IOTYPE_UNICODEPATH, the specification must contain UCS-2 encoded Unicode characters.

  When your application receives this callback, the contents of the buffer pointed to by pSpec are undefined. A specification must be defined by your application if this callback returns SCCERR_OK.
- pExportData: Pointer to data specific to the individual export. In this case, always a pointer to either an EXURLFILEIOCALLBACKDATA structure or an EXURLFILEIOCALLBACKDATAW structure. The EXURLFILEIOCALLBACKDATAW struct is only used when the SCCOPT_UNICODECALLBACKSTR option is set to TRUE. These two structures are defined in Section 9.1.1.1, "EXURLFILEIOCALLBACKDATA / EXURLFILEIOCALLBACKDATAW Structures." Set by caller.
- pTemplateName: NULL

9.1.1.1 EXURLFILEIOCALLBACKDATA / EXURLFILEIOCALLBACKDATAW Structures

The EXURLFILEIOCALLBACKDATA and EXURLFILEIOCALLBACKDATAW structures are defined as follows:

```
typedef struct EXURLFILEIOCALLBACKDATAtag
{
   VTDWORD   dwSize;
   VTBYTE    szURLString[VT_MAX_URL];
   VTDWORD   dwFileID;
} EXURLFILEIOCALLBACKDATA;

typedef struct EXURLFILEIOCALLBACKDATAWtag
{
   VTDWORD   dwSize;
   VTWORD    wzURLString[VT_MAX_URL];
   VTDWORD   dwFileID;
} EXURLFILEIOCALLBACKDATAW;
```

- dwSize: Set to sizeof(EXURLFILEIOCALLBACKDATA) or sizeof(EXURLFILEIOCALLBACKDATAW).
- szURLString / wzURLString: This parameter can be set by the developer to a new URL that references the newly created file. This parameter is optional unless the pSpec provided by the developer points to something that cannot be used as a URL (as when using redirected IO, for example). In that case, this parameter must be set.

  This string is written into any output file that needs to reference the newly created file, with appropriate conversions between single and double byte output. Because this parameter is a URL, it is assumed to be URL encoded. When used in conjunction with dwSpecType and pSpec, this parameter can be used to generate almost any structure or location for the output files, including things like writing the output files into a database and then using a CGI mechanism to retrieve them.

  The current size limitation is 2048 characters. If the size exceeds this limit, the URL will be truncated and rendered useless.
- dwFileID: Set by the product. This is used as a unique identifier for each output file generated. It may be used for an OEM-specific purpose.

Return Value

- SCCERR_OK: dwSpecType, pSpec and szURLString (or wzURLString) have been populated with valid values.
- SCCERR_NOTHANDLED: Default naming should be used.
- SCCERR_FILEOPENFAILED: Some error was encountered creating a new output.

9.1.2 EX_CALLBACK_ID_NEWFILEINFO

This informational callback is made just after each new file has been created. Like the EX_CALLBACK_ID_CREATENEWFILE callback, the pExportData parameter points to an EXURLFILEIOCALLBACKDATA or an EXURLFILEIOCALLBACKDATW structure, but in this case the structure should be treated as read-only and the dwSpecType, pSpec and szURLString (or wzURLString) will be filled in.

This callback occurs for every new file. If the developer has used the EX_CALLBACK_ID_CREATENEWFILE notification to change the location of (or to set up redirected IO

for) the new file, the data structure echoes back the information set by the developer during the EX_CALLBACK_ID_CREATENEWFILE callback.

Return Value

Must be either SCCERR_OK or SCCERR_NOTHANDLED. Return value is currently ignored.

9.1.3 EX_CALLBACK_ID_PAGECOUNT

Web View Export uses this callback message to return a count of all of the output pages produced during an export operation. This count reflects the number of pages created by Oracle Outside In's processing of the input document, which in some cases may differ slightly from the number of pages as seen in the document's original application.

This callback occurs during the execution of EXRunExport.

Data Type

VTDWORD

10

Web View Export Options

Options are parameters affecting the behavior of an export or transformation. This chapter presents the C/C++ options relevant to the Web View Export product.

10.1 Web View Export C/C++ Options

Options are set using the DASetOption call. It is recommended that developers familiarize themselves with all of the options available.

Options may be Local, in which case they only affect the handle for which they are set, or Global, in which case they automatically affect all handles associated with the hDoc and must be set before the call to DAOpenDocument.

While default values are provided, users are encouraged to set all options for a number of reasons. In some cases, the default values were chosen to provide backwards compatibility. In other cases, the default values were chosen arbitrarily from a range of possibilities.

10.1.1 Character Mapping

This section discusses character mapping options.

10.1.1.1 SCCOPT_DEFAULTINPUTCHARSET

This option is used in cases where Oracle Outside In cannot determine the character set used to encode the text of an input file. When all other means of determining the file's character set are exhausted, Oracle Outside In will assume that an input document is encoded in the character set specified by this option. This is most often used when reading plain-text files, but may also be used when reading HTML or PDF files. The possible character sets are listed in charsets.h.

When "extended test for text" is enabled (see Section 10.1.2.2, "SCCOPT_FIFLAGS"), this option will still apply to plain-text input files that are not identified as EBCDIC or Unicode.

This option supersedes the SCCOPT_FALLBACKFORMAT option for selecting the character set assumed for plain-text files. For backwards compatibility, use of deprecated character-set -related values is still currently supported for SCCOPT_FALLBACKFORMAT, though internally such values will be translated into equivalent values for the SCCOPT_DEFAULTINPUTCHARSET. As a result, if an application were to set both options, the last such value set for either option will be the value that takes effect.

Data Type
VTDWORD

Default
- Windows Code Page 1252 on Windows and ISO 8859-1 (Latin 1) on UNIX

10.1.1.2 SCCOPT_UNMAPPABLECHAR
This option selects the character used when a character cannot be found in the output character set. This option takes the Unicode value for the replacement character.

Data Type
VTWORD

Default
- 0x002a = "*"

10.1.2 Input Handling
This section discusses input handling options.

10.1.2.1 SCCOPT_FALLBACKFORMAT
This option controls how files are handled when their specific application type cannot be determined. This normally affects all plain-text files, because plain-text files are generally identified by process of elimination, for example, when a file isn't identified as having been created by a known application, it is treated as a plain-text file.

It is recommended that FI_NONE be set to prevent Web View Export from exporting unidentified binary files as though they were text, which could generate many pages of "garbage" output.

This option must be set for an hDoc before any subhandle has been created for that hDoc.

A number of values that were formerly allowed for this option have been deprecated. Specifically, the values that select specific plain-text character sets are no longer to be used. Instead, applications should use the SCCOPT_DEFAULTINPUTCHARSET option for such functionality.

Data Type
VTDWORD

Default
- FI_TEXT

10.1.2.2 SCCOPT_FIFLAGS
This option affects how an input file's internal format (application type) is identified when the file is first opened by the Oracle Outside In technology. When the extended test flag is in effect, and an input file is identified as being either 7-bit ASCII, EBCDIC, or Unicode, the file's contents will be interpreted as such by the export process.

The extended test is optional because it requires extra processing and cannot guarantee complete accuracy (which would require the inspection of every single byte in a file to eliminate false positives.)

Data Type
VTDWORD

Default
- SCCUT_FI_EXTENDEDTEST: The technology will attempt an extra test after the file is first opened to see if it is 7-bit text or EBCDIC.

10.1.2.3 SCCOPT_FORMATFLAGS
This option allows the developer to set flags that enable options that span multiple export products.

Data Type
VTDWORD

Default
0: All flags turned off

10.1.2.4 SCCOPT_SYSTEMFLAGS
This option controls a number of miscellaneous interactions between the developer and the Outside In Technology.

Data Type
VTDWORD

Default
0

10.1.2.5 SCCOPT_IGNORE_PASSWORD
This option can disable the password verification of files where the contents can be processed without validation of the password. If this option is not set, the filter should prompt for a password if it handles password-protected files.

As of Release 8.4.0, only the PST and MDB Filters support this option.

Data Type
VTBOOL

Default
FALSE

10.1.2.6 SCCOPT_LOTUSNOTESDIRECTORY
This option allows the developer to specify the location of a Lotus Notes or Domino installation for use by the NSF filter. A valid Lotus installation directory must contain the file nnotes.dll.

> **Note:** Please see section 2.1.1 for NSF support on Win x86-32 or Win x86-64 or section 3.1.1 for NSF support on Linux x86-32 or Solaris Sparc 32.

Data Type
VTLPBYTE

Default
If this option isn't set, then OIT will first attempt to load the Lotus library according to the operating system's PATH environment variable, and then attempt to find and load the Lotus library as indicated in HKEY_CLASSES_ROOT\Notes.Link.

10.1.2.7 SCCOPT_PDF_FILTER_REORDER_BIDI
This option controls whether or not the PDF filter will attempt to reorder bidirectional text runs so that the output is in standard logical order as used by the Unicode 2.0 and later specification. This additional processing will result in slower filter performance according to the amount of bidirectional data in the file.

Data Type
VTDWORD

Default
SCCUT_FILTER_STANDARD_BIDI

10.1.2.8 SCCOPT_REORDERMETHOD
This option controls how the technology reorders bidirectional text.

Data Type
VTDWORD

10.1.2.9 SCCOPT_TIMEZONE
This option allows the user to define an offset to GMT that will be applied during date formatting, allowing date values to be displayed in a selectable time zone. This option affects the formatting of numbers that have been defined as date values. This option will not affect dates that are stored as text.

> **Note:** This option does not apply for spreadsheet files.

Data Type
VTLONG

Default
- 0: GMT time

10.1.2.10 SCCOPT_HTML_COND_COMMENT_MODE
Some HTML includes a special type of comment that will be read by particular versions of browsers or other products. This option allows you to control which of those comments are included in the output.

Data Type
VTDWORD

Default

- NONE: Don't output any conditional comment

10.1.2.11 SCCOPT_ARCFULLPATH

In the Viewer and rendering products, this option tells the archive display engine to show the full path to a node in the szNode field in response to a SCCVW_ GETTREENODE message. It also causes the name fields in DAGetTreeRecord and DAGetObjectInfo to contain the full path instead of just the archive node name.

Data Type

VTBOOL

Default

FALSE

10.1.3 Graphics

This section discusses graphics options.

10.1.3.1 SCCOPT_GRAPHIC_OUTPUTDPI

This option allows the user to specify the output graphics device's resolution in DPI and only applies to images embedded in a PDF whose size is specified in physical units (in/cm). For example, consider a 1" square, 100 DPI graphic that is to be rendered on a 50 DPI device (SCCOPT_GRAPHIC_OUTPUTDPI is set to 50). In this case, the size of the resulting PDF will be 50 x 50 pixels.

In addition, the special #define of SCCGRAPHIC_MAINTAIN_IMAGE_DPI, which is defined as 0, can be used to suppress any dimensional changes to an image. In other words, a 1" square, 100 DPI graphic will be converted to an image that is 100 x 100 pixels in size. This value indicates that the DPI of the output device is not important. It extracts the maximum resolution from the input image with the smallest exported image size.

Setting this option to SCCGRAPHIC_MAINTAIN_IMAGE_DPI may result in the creation of extremely large images. Be aware that there may be limitations in the system running this technology that could result in undesirably large bandwidth consumption or an error message. Additionally, an out of memory error message will be generated if system memory is insufficient to handle a particularly large image.

Also note that the SCCGRAPHIC_MAINTAIN_IMAGE_DPI setting will force the technology to use the DPI settings already present in raster images, but for all other content the resolution used internally by Web View Export will be in effect.

For some output graphic types, there may be a discrepancy between the value set by this option and the DPI value reported by some graphics applications. The discrepancy occurs when the output format uses metric units (DPM, or dots per meter) instead of English units (DPI, or dots per inch). Depending on how the graphics application performs rounding on meters to inches conversions, the DPI value reported may be 1 unit more than expected.

Data Type

VTDWORD

Default

- SCCGRAPHIC_DEFAULT_OUTPUT_DPI: Currently defined to be 72 dots per inch.

10.1.3.2 SCCOPT_GRAPHIC_SIZEMETHOD

This option determines the method used to size graphics. The developer can choose among three methods, each of which involves some degree of trade off between the quality of the resulting image and speed of conversion.

Using the quick sizing option results in the fastest conversion of color graphics, though the quality of the converted graphic will be somewhat degraded. The smooth sizing option results in a more accurate representation of the original graphic, as it uses anti-aliasing. Antialiased images may appear smoother and can be easier to read, but rendering when this option is set will require additional processing time. The grayscale only option also uses antialiasing, but only for grayscale graphics, and the quick sizing option for any color graphics.

The smooth sizing option does not work on images which have a width or height of more than 4096 pixels.

Data Type

VTDWORD

Default

SCCGRAPHIC_SMOOTHSIZING

10.1.4 Page Rendering

This section discusses page rendering options.

10.1.4.1 SCCOPT_DEFAULTPAGESIZE

This option allows the developer to specify the size of each page in the generated PDF output file. The size may be specified in inches, points, centimeters or picas. This option is only valid when SCCOPT_USEDOCPAGESETTINGS is set to FALSE.

1 inch = 6 picas = 72 points = ~ 2.54 cm

Data Type

DEFAULTPAGESIZE Structure

Default

8.5 inches by 11 inches

0.0.0.0.1 DEFAULTPAGESIZE Structure

```
typedef struct DEFAULTPAGESIZEtag
{
   VTFLOAT dwHeight;
   VTFLOAT dwWidth;
   VTDWORD wUnits;
}DEFAULTPAGESIZE,  *LPDEFAULTPAGESIZE;
```

Parameters

Note: You must define a value for both dwHeight and dwWidth in wUnits. If you define only height or only width, the image is not scaled.

- dwHeight: Height of the page. Default is 11 inches.
- dwWidth: Width of the page. Default is 8.5 inches.
- wUnits: One of the following (SCCGRAPHIC_INCHES is the default):
  - SCCGRAPHIC_INCHES
  - SCCGRAPHIC_POINTS
  - SCCGRAPHIC_CENTIMETERS
  - SCCGRAPHIC_PICAS

10.1.4.2 SCCOPT_DEFAULTPRINTMARGINS

This option specifies the top, left, bottom and right margins in twips from the edges of the page. For instance, setting all the values to 1440 creates a 1-inch margin on all sides. Page margins will only be applied when formatting word processing, database and spreadsheet files.

This option is overridden when the SCCOPT_USEDOCPAGESETTINGS option is set to TRUE and print margins are specified in the input document.

This option does not affect the output of bitmap, presentation, vector or archive files.

Data Type

The SCCVWPRINTMARGINS structure.

0.0.0.0.2 SCCVWPRINTMARGINS Structure This structure is used by the SCCOPT_DEFAULTPRINTMARGINS option to specify margin settings.

SCCVWPRINTMARGINS is a C data structure defined in sccvw.h as follows:

```
typedef struct SCCVWPRINTMARGINStag
   {
   VTDWORD  dwTop;
   VTDWORD  dwBottom;
   VTDWORD  dwLeft;
   VTDWORD  dwRight;
} SCCVWPRINTMARGINS,  * PSCCVWPRINTMARGINS;
```

Parameters

- dwTop: Margin from the top edge of the page (in twips). Default is 1 inch.
- dwBottom: Margin from the bottom edge of the page (in twips). Default is 1 inch.
- dwLeft: Margin from the left edge of the page (in twips). Default is 1 inch.
- dwRight: Margin from the right edge of the page (in twips). Default is 1 inch.

10.1.4.3 SCCOPT_PRINTENDPAGE

This option indicates the page that rendering should end on. It is only valid if the option SCCOPT_WHATTOPRINT has the value SCCVW_PRINT_PAGERANGE.

Note that page range settings are one-based and inclusive. Therefore, specifying a range with SCCOPT_PRINTENDPAGE equal to 5 and SCCOPT_PRINTSTARTPAGE equal to 3 would export any of the three pages that follow, if they exist: 3, 4 and 5.

Data Type
VTDWORD

Default
- 0: The last page at the end of the document.

10.1.4.4 SCCOPT_PRINTSTARTPAGE
This option indicates the page rendering should start on. It is only valid if the option SCCOPT_WHATTOPRINT has the value SCCVW_PRINT_PAGERANGE.

Note that page range settings are one-based and inclusive. Therefore, specifying a range with SCCOPT_PRINTENDPAGE equal to 5 and SCCOPT_PRINTSTARTPAGE equal to 3 would export any of the three pages that follow, if they exist: 3, 4 and 5.

Data Type
VTDWORD

Default
- 0: Printing will begin with the first page of the document.

10.1.4.5 SCCOPT_USEDOCPAGESETTINGS
This option is used to select the document's page layout information when rendering.

If TRUE, the document's native (or author selected) page margins, paper size, page scaling and page orientation are used when available from the filter.

The values of the SCCOPT_DEFAULTPAGESIZE and SCCOPT_DEFAULTPRINTMARGINS options are overridden if this option is set to TRUE and the properties associated with those options are specified in the input document. Additionally, print area and page breaks in spreadsheet documents are ignored unless this option is set to TRUE.

If FALSE, the page margins, size, orientation and scaling are set to specific values rather than those in the native document. The page size is forced to 8 1/2" x 11" in portrait orientation, but this may be changed by setting the SCCOPT_DEFAULTPAGESIZE option. The margins are forced 1" all around, but may be changed by setting the SCCOPT_DEFAULTPRINTMARGINS option. The scaling for the document will be set to 100%, although this may be changed by setting any of the various scaling options.

It should be noted that this option also affects page orientation for both input spreadsheets and word processing documents.

Data Type
VTBOOL

Default
TRUE

10.1.4.6 SCCOPT_WHATTOPRINT
This option indicates whether the whole file or a selected range of pages should be rendered.

Data Type
VTDWORD

Default
SCCVW_PRINT_ALLPAGES

10.1.4.7 SCCOPT_NUMBERFORMAT
This option is used to control the formatting of numbers. It is useful for setting environment dependent variables related to international support. The default values are retrieved from the operating system for the Windows platform, and are set to logical U.S. defaults on all other platforms.

Data Type
SCCVWNUMBERFORMAT and SCCVWNUMBERFORMAT775 structures

0.0.0.0.3 SCCVWNUMBERFORMAT775 and SCCVWNUMBERFORMAT Structures These structures are used to set the SCCID_NUMBERFORMAT option. The fields of the structures allow the developer to control variables related to international support. Please note that the SCCVWNUMBERFORMAT775 structure always assumes 2-digit year data, whereas the SCCVWNUMBERFORMAT structure allows for both 2- and 4-digit year data.

These are C data structures defined in sccvw.h as follows:

```
typedef struct SCCVWNUMBERFORMAT775tag
   {
   VTTCHAR   cDecimalSep;
   VTTCHAR   cThousandSep;
   VTTCHAR   cDateSep;
   VTTCHAR   cTimeSep;
   VTTCHAR   szCurrencySymbol[8];
   VTTCHAR   szAM[8];
   VTTCHAR   szPM[8];
   VTDWORD   dwNumBytesAM;
   VTDWORD   dwNumBytesPM;
   VTWORD    wCurrencyPosition;
   VTWORD    wShortDateOrder;
   } SCCVWNUMBERFORMAT775,  * PSCCVWNUMBERFORMAT775;

typedef struct SCCVWNUMBERFORMATtag
   {
   VTTCHAR   cDecimalSep;
   VTTCHAR   cThousandSep;
   VTTCHAR   cDateSep;
   VTTCHAR   cTimeSep;
   VTTCHAR   szCurrencySymbol[8];
   VTTCHAR   szAM[8];
   VTTCHAR   szPM[8];
   VTDWORD    dwNumBytesAM
   VTDWORD   dwNumBytesPM;
   VTWORD    wCurrencyPosition
   VTWORD    wShortDateOrder;
   VTWORD    wShortDateYearDigits;
   VTWORD    wShortDateMonthDigits;
   VTWORD    wShortDateDayDigits;
   VTWORD    wShortDateFlags;
   } SCCVWNUMBERFORMAT,  * PSCCVWNUMBERFORMAT;
```

Parameters

- cDecimalSep: The character used for the decimal separator when formatting currency.
- cThousandSep: The character used for the thousands separator when formatting currency.
- cDateSep: The character used to separate years, months, and days when formatting dates. This option only works on variable formats. For example, only one of the several date formats in Microsoft Excel is variable.
- cTimeSep: The character used to separate hours, minutes, and seconds when formatting times. This option only works on variable formats. For example, only one of the several time formats in Microsoft Excel is variable.
- szCurrencySymbol: The string used for the currency symbol when formatting currency.
- szAM: The string used to indicate "AM" when formatting times.
- szPM: The string used to indicate "PM" when formatting times.
- dwNumBytesAM: Number of bytes of the string stored in szAM.
- dwNumBytesPM: Number of bytes of the string stored in szPM.
- wCurrencyPosition: Flags that indicate the positioning of the currency symbol when formatting currency. Only six specific filters are supported: SOC6, WG2, WK4, WK6, WPW, and VISO.
  - SCCVW_CURRENCY_LEADS: The currency symbol is placed before the amount.
  - SCCVW_CURRENCY_TRAILS: The currency symbol is placed after the amount.
  - SCCVW_CURRENCY_SPACE: A space is placed between the currency and the amount.
  - SCCVW_CURRENCY_NOSPACE: A space is not placed between the currency and the amount.
- wShortDateOrder: Indicates the order used when formatting short dates (numeric dates). This option only works on variable formats. For example, only one of the several date formats in Microsoft Excel is variable. One of the following:
  - SCCVW_DATEORDER_MDY: Month, Day, Year
  - SCCVW_DATEORDER_DMY: Day, Month, Year
  - SCCVW_DATEORDER_YMD: Year, Month, Date
- wShortDateYearDigits: This parameter is specific to the SCCVWNUMBERFORMAT structure. This is the number of digits in the year as specified by the Windows registry entry sShortDate. This option only works on variable formats. For example, only one of the several date formats in Microsoft Excel is variable.
- wShortDateMonthDigits: This parameter is specific to the SCCVWNUMBERFORMAT structure. This is the number of digits in the month as specified by the Windows registry entry sShortDate.
- wShortDateDayDigits: This parameter is specific to the SCCVWNUMBERFORMAT structure. This is the number of digits in the day as specified by the Windows registry entry sShortDate.

- wShortDateFlags: This parameter is specific to the SCCVWNUMBERFORMAT structure. It is reserved for internal use.

10.1.5 Font Rendering

This section discusses font rendering options.

10.1.5.1 SCCOPT_DEFAULTPRINTFONT

This is an advanced option that casual users of Web View Export may ignore.

This option sets the font to use when the chunker-specified font is either excluded or is not available on the system. It is also the font used when the font in the source file is not available on the system performing the conversion.

Data Type

SCCVWFONTSPECstructure

0.0.0.0.4 SCCVWFONTSPEC Structure This structure is used by various options to specify a font.

SCCVWFONTSPEC is a C data structure defined in sccvw.h as follows:

```
typedef struct
   {
   VTTCHAR  szFace[40];
   VTWORD   wHeight;
   VTWORD   wAttr;
   VTWORD   wType;
   } SCCVWFONTSPEC,  * LPSCCVWFONTSPEC;
```

Parameters

- szFace: The name of the font. For example, "Helvetica Compressed." The default is "Arial", however this default is constrained by the fonts available on the system.
- wHeight: Size of the font in half points. For example, a value of 24 will produce a 12-point font. This size is only applied when the font size is not known. The default is 10-point, however this default is constrained by the font sizes available on the system.
- wAttr: The attributes of the font. This parameter is used primarily by the Oracle Outside In Viewer Technology and is currently ignored by Web View Export.
- wType: Should be set to 0.

10.1.5.2 SCCOPT_FONTDIRECTORY

This option allows the developer to specify one or more font directories where fonts are located for use by Web View Export. If multiple font directories are specified, they should be delimited by a colon on Linux and UNIX systems and a semi-colon on Windows systems.

This option must be set prior to performing any exports. Please note that Web View Export supports single TrueType fonts (*.ttf, *.TTF) and TrueType collections (*.ttc, *.TTC), not Windows bitmap fonts (*.fon, *.FON), or any other type of font. Also, Web View Export does not require case-sensitive font filenames on UNIX systems.

Data Type

VTLPBYTE

Default

NONE - the option must be set.

10.1.6 Web View Export Options

This section discusses Web View Export options.

10.1.6.1 SCCOPT_OUTPUT_STRUCTURE

This option enables "Ajax mode" and "Streaming mode" in web view output files. Enabling these modes directly affects the structure of the output files created by Web View Export.

- When this option is set to SCCWV_STRUCTURE_AJAX_CHUNKED or SCCWV_STRUCTURE_AJAX_STREAMING, the output files that comprise the web view of a document will use Ajax techniques to load portions of the web view on demand, in response to browser-based requests. This generally means that the export process will produce a higher number of smaller output files.
- When the option is set to SCCWV_STRUCTURE_AJAX_STREAMING, client side viewing may begin as soon as the primary output file is created, and before the export has completed. This reduces the wait time when exporting very large documents.
- When this option is SCCWV_STRUCTURE_FLAT, the entire document will be exported in a single operation to fewer, larger output files.

There are other practical differences implied by Ajax mode.

When Ajax Chunked Mode or Ajax Streaming Mode is Enabled

- HTML content is spread among multiple output pages.
- Document internal data is exported in JSON format, and stored in multiple output files.
- Documents produced in Ajax mode can be embedded into another page with the OIT.view.load Javascript API method.
- Documents produced in Ajax mode generally cannot be opened by a browser directly from the local file system (e.g., from a file:/// URL.) This is due to security features that restrict most browsers to using Ajax loading methods only on files accessed from web servers.

When Ajax Streaming Mode is Enabled

- The browser may begin viewing the document before the entire output has been produced. (The primary output file may be served once its NEWFILEINFO callback message is sent. See documentation of Export callback function for details.)
- Font subsetting is not available. If SCCOPT_FONT_REFERENCE_METHOD is set to SCCFONTS_REFERENCE_EXPORTED, it will revert to the default of SCCFONTS_REFERENCE_BY_NAME.

When Ajax Mode is Disabled

- HTML content is exported in a single HTML file.
- Document internal data is exported in a single Javascript file.

- The web view output may not be viewed until the entire export operation is complete.
- Documents produced without Ajax mode enabled may not be loaded via OIT.view.load.
- Documents produced without Ajax mode enabled may be opened by a browser directly from the local file system (possibly with a security warning regarding Javascript.).

Data Type

DWORD enumeration; Must be one of SCCWV_STRUCTURE_FLAT, SCCWV_STRUCTURE_AJAX_CHUNKED, SCCWV_STRUCTURE_AJAX_STREAMING

Default

SCCWV_STRUCTURE_FLAT

10.1.6.2 SCCOPT_URLPATH_RESOURCES

String option for specifying the base URL for Web View Export scripts, css files and other resources that do not vary with the converted file. The value of this option is used as a prefix for references to those resources in the primary output file.

For example, if the resources' base URL is "/oit/", the script reference in the output file would look like this:

```
<script src="/oit/oit.js"></script>
```

By default, this value is empty, meaning all resources must be present in the same directory as the generated output.

Data Type

utf-8 encoded string

Default

""

10.1.6.3 SCCOPT_URLPATH_OUTPUT

String option for specifying the base URL for output files generated during the export process. The value of this option is used as a prefix for references to those resources in the primary output file.

If the output base URL is "/exports/", an image tag in an exported document would look like this:

```
<img src="/exports/doc01.doc.jpg" />
```

and URLs to the exported files within content.js would be of this form:

```
{id:"page.1",name:"Page
1",type:"page",oid:"overlay.1",w:"816px",h:"1056px",src:"/exports/output.3.html"},
```

By default, the output file base URL is empty, and all output will be specified without path information.

Data Type

utf-8 encoded string

Default
""

10.1.6.4 SCCOPT_EXTERNAL_STYLESHEET
String option for specifying the url of a user supplied stylesheet to be included in the output html file. Values specified through this option will appear in output HTML files as linked stylesheets.

Example:

```
char * pCustomStyles = "/some/path/myStylesheet.css";
DAAddOptionItem( hExport, SCCOPT_POST_LIBRARY_SCRIPT, pInitScript,
strlen(pCustomStyles), NULL );
```

which results in the following contents in the EXH5 output:

```
<script src="oit.css">
<link rel="stylesheet" href="/some/path/myStylesheet.css"/>
```

10.1.6.5 SCCOPT_POST_LIBRARY_SCRIPT
String option specifying the relative URL of the customer's initialization script. Scripts specified via this option will load after the Outside In libraries, and may access the Outside In javascript API. This option may be called multiple times; the scripts will be referenced in the same order as the calls to set this option.

Example:

```
char * pInitScript = "/config/oitinit.js";
DWORD optionId;
DAAddOptionItem( hExport, SCCOPT_POST_LIBRARY_SCRIPT, pInitScript,
strlen(pInitScript), &optionId );
```

which results in the following contents in the EXH5 output:

```
<script src="oit.js">
<script src="/config/oitinit.js">
```

Data Type
utf-8 encoded URL string

Default
None

10.1.6.6 SCCOPT_PRE_LIBRARY_SCRIPT
String option specifying the URL of a script file to be referenced via a <script> tag in the output file. Scripts specified via this option will load before the Outside In libraries. This option may be called multiple times; the scripts will be referenced in the same order as the calls to set this option.

Example:

```
char * pJQlib = "/jQuery/jQuery-min.js";
DAAddOptionItem( hExport, SCCOPT_PRE_LIBRARY_SCRIPT, pJQlib, strlen(pJQlib), NULL
);

char * pHelperLib = "/shims/customstuff.js";
DAAddOptionItem( hExport, SCCOPT_PRE_LIBRARY_SCRIPT, pHelperLib,
strlen(pHelperLib), NULL );
```

which results in the following contents in the EXH5 output:

```
<script src="/jQuery/jQuery-min.js">
<script src="/shims/customstuff.js">
<script src="oit.js">
```

Data Type

utf-8 encoded URL string

Default

None

10.1.6.7 SCCOPT_OUTPUT_RAWTEXT

Boolean option controlling whether raw text output is generated on the server side. This option must be enabled in order to enable in-browser searching via the Outside In API. This does not affect the search capabilities of the browser itself.

Raw text generation is enabled by default.

Data Type

BOOLEAN

Default

TRUE

10.1.6.8 SCCOPT_FONT_PERMISSIONS_MODE

This option controls the use of "embedded" fonts (see SCCOPT_FONT_REFERENCE_METHOD) in the Web View Export output; specifically, whether or not a downloadable (web font) version of the font should be made available to Web View Export output, based on the licensing restrictions indicated within the font itself.

Data Type

DWORD flags; value must be any of the following ORed together: SCCFONTS_DEFAULT_PERMISSIONS, SCCFONTS_REQUIRE_INSTALLABLE, SCCFONTS_REQUIRE_PREVIEW_PRINT, SCCFONTS_REQUIRE_EDITABLE

Default

SCCFONTS_DEFAULT_PERMISSIONS

Values

- SCCFONTS_DEFAULT_PERMISSIONS: Normally restricted per the TrueType specification. This requires that the font declare itself as capable of being embedded.
- SCCFONTS_REQUIRE_INSTALLABLE: Font selection is tightly restricted (a la Internet Explorer 9/10). This requires that the font declare itself as capable of being installed.
- SCCFONTS_REQUIRE_PREVIEW_PRINT: This requires that the font declare itself as capable of preview and print. When this bit is set, the font may be embedded, and temporarily loaded on the remote system. Documents containing Preview & Print fonts must be opened as read-only; no edits can be applied to the document.

- SCCFONTS_REQUIRE_EDITABLE: This requires that the font declare itself as capable being edited. When this bit is set, the font may be embedded but must only be installed temporarily on other systems. In contrast to Preview & Print fonts, documents containing Editable fonts may be opened for reading, editing is permitted, and changes may be saved.

10.1.6.9 SCCOPT_FONT_REFERENCE_METHOD

This option controls the way embedded fonts are presented (or not) in the HTML5 output.

Data Type

DWORD enumeration; value must be one of the following: SCCFONTS_REFERENCE_BY_NAME, SCCFONTS_REFERENCE_EXPORTED, or SCCFONTS_REFERENCE_BY_BASE_URL.

Default

: SCCFONTS_REFERENCE_BY_NAME

Values

- SCCFONTS_REFERENCE_BY_NAME: The font is referenced in the output file by name only. For example: "Baskerville".
- SCCFONTS_REFERENCE_EXPORTED: The font is exported during document conversion, and referenced in the output by file name. For example: "ttf002.ttf".
- SCCFONTS_REFERENCE_BY_BASE_URL: The full font is referenced in the output using the font file name and the base URL supplied by the SCCOPT_FONT_BASE_URL option. For example: "http://baseurl.com/Baskerville.ttf". If SCCOPT_FONT_BASE_URL is not specified then this option reverts to the default (SCCFONTS_REFERENCE_BY_NAME).

Option Effects in CSS

SCCFONTS_REFERENCE_BY_NAME: Reference by family name only

```
.oit-span-6 {
    font: 0.65pt "Arial", "Helvtetica Neue", Helvetica, sans-serif;
}
```

SCCFONTS_REFERENCE_EXPORTED: Reference exported font.

```
@font-face {
    font-family:  f101;
    src: url('m2.f101.ttf');
}
.oit-span-6 {
    font: 0.65pt "Arial", f101, "Helvtetica Neue", Helvetica, sans-serif;
}
```

SCCFONTS_REFERENCE_BY_BASE_URL: Reference original font with fontBaseUrl, if specified. Otherwise default to SCCFONTS_REFERENCE_BY_NAME.

```
@font-face {
    font-family:  f101;
    src: url('fontBaseUrl/Arial.ttf');
}
.oit-span-6 {
    font: 0.65pt "Arial", f101, "Helvtetica Neue", Helvetica, sans-serif;
```

```
}
```

10.1.6.10 SCCOPT_FONT_BASE_URL

This option allows the developer to specify the base URL used in the Web View Export output when referencing a non-exported font as a downloadable font.

(Exported fonts will be placed in the exported output location.)

Data Type

utf-8 encoded URL string

Default

None

10.1.6.11 SCCOPT_EMAIL_ATTACHMENT_HANDLING

This option determines whether email attachments are converted or extracted.

Data Type

DWORD enumeration; value must be one of the following: SCCOPT_EXPORT, SCCOPT_EXPORT_RECURSIVE, or SCCOPT_EXTRACT

Default

None

Values

- SCCOPT_EXPORT: The attachments to this email will be converted.
- SCCOPT_EXPORT_RECURSIVE: The attachments to this email will be converted; if any are emails, they will also have their attachments converted, and so on.
- SCCOPT_EXTRACT: The attachments will be extracted in their native form, stored alongside the Web View Export output files.

10.1.6.12 SCCOPT_STAMP_IMAGE_URL

Image stamp annotations are bitmap images that are "stamped" onto the rendition of a file generated by Outside In. The source of the image may be a PNG, JPG, or GIF image. They may be considered special type of rectangular annotation.

In the Export API, image stamps are applied in the following way:

- Names are defined for the source images. In the C API, this is accomplished via calling DAAddOptionItem as many times as necessary, with the option SCCOPT_STAMP_IMAGE_URL (for Web View Export output files) or SCCOPT_STAMP_IMAGE_FILE (for Image Export output files.)
- A stamp is applied by specifying its destination rectangle and the name of its source image. In the C API, this is accomplished by populating an EXANNOSTAMP data structure and specifying it as a parameter to EXAddStampAnnotation.

The EXANNOSTAMPIMAGE data structure has the following definition:

```
typedef struct EXANNOSTAMPIMAGE
{
  VTLPCTSTR   szStampName;
  VTLPVOID    pStampImage;
```

```
  VTDWORD     dwSpecType;
} EXANNOSTAMPIMAGE ;
```

- szStampName is a null-terminated UTF8 string that specifies the name of a stamp
- pStampImage is the IO spec of the stamp image.
- dwSpecType is the type of IO spec provided by pStampImage

When setting the SCCOPT_STAMP_IMAGE_URL option, pStampImage must be a null-terminated UTF8 encoded URL to a PNG, JPG, or GIF image. dwSpecType should be set to IOTYPE_URL.

If the name specified by szStampName has already been used in a previous call to SCCOPT_STAMP_IMAGE_URL, its URL will be updated with the new value.

Data Type

pointer to EXANNOSTAMPIMAGE structure

Default

none

10.1.6.13 SCCOPT_WV_LIBRARY_NAME

String option specifying the file name of the Web View Javascript library, which will be written into a <script> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.js" will be used.

This option allows the Web View Javascript library to be renamed for versioning or other purposes. This option works in combination with the resource path option.

Data Type

utf-8 encoded string

Default

"oit.js"

10.1.6.14 SCCOPT_WV_STYLESHEET_NAME

String option specifying the file name of the Web View stylesheet, which will be written into a <link> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.css" will be used.

This option allows the Web View stylesheet to be renamed for versioning or other purposes. This option works in combination with the resource path option.

Data Type

utf-8 encoded string

Default

"oit.css"

Part III

Using the Javascript API

This section provides details about using the SDK with the Javascript API.

Part III contains the following chapter:

11

Web View Export Javascript API

The output produced by Web View Export (EXH5) includes references to Javascript files that provide functionality within the exported content. These files also present an API for users to implement customized integration and interaction with that output.

This appendix contains the following sections:

11.1 The Global OIT Object

Web View Export's API functions will be implemented as methods of a global object named, as of now, OIT. Similar to the $ object used by jQuery, all interaction by 3rd party scripts with Web View Export content will be through this object. This will prevent name collisions between any of our functions and those in scripts written by other parties.

11.2 Client-side Display Options

Some aspects of the Web View Export display will be controllable through the OIT.document.setOptions() function. Note that these will not affect conversion-time behavior.

11.3 Client-side Control

It's very possible that a consumer of Web View Export output, having implemented a web application with their own user interface elements, might not want to rely on controls provided by us (e.g. navigation controls, scrollbars, statusbar, etc.).

Web View Export will provide a basic API for control of the navigation of its content, through which a customer may implement their own interface to control the users' interaction with content rendered by Web View Export.

These methods will be accessed from objects under the OIT object:

- **OIT.archive**: This object allows the customer to listen to events related to archive files.
- **OIT.attachments**: This object provides access to file attachments.
- **OIT.document**: Methods of this object control zooming, searching, and selection of the content.
- **OIT.highlights**: Management of highlighted text, cell ranges, or rect ranges.
- **OIT.hyperlinks**: This object allows the customer to listen to events related to hyperlinks.
- **OIT.local**: Localized user interface strings.
- **OIT.toolbars**: This object provides control over the toolbars.
- **OIT.pages**: Methods of this object control navigation through the pages of the document.
- **OIT.view**: This object provides control over the view.

11.4 Ranges

Specialized ranges are used throughout the Web View Export API.

11.4.1 TextRange

A TextRange is a specialized range that tracks a range of text by ACC (actual character count).

Field	Type	Description
type	String	The range type: "text".
acc	Number	The starting ACC for the range.
accend	Number	The ending ACC for the range.

11.4.2 CellRange

A CellRange is a specialized range that tracks a cell or range of cells in a spreadsheet or database table.

Field	Type	Description
type	String	The range type: "cell".
page	Number	The 0-based index of the page containing the range.

Field	Type	Description
section	Number	The 0-based index of the section containing the range (includes hidden pages)
c1	Number	The 0-based index of the starting column for the range.
r1	Number	The 0-based index of the starting row for the range.
c2	Number	The 0-based index of the ending column for the range. For single column ranges, this will be equal to c1.
r2	Number	The 0-based index of the ending row for the range. For single row ranges, this will be equal to r1.

11.4.3 RectRange

A RectRange is a specialized range that tracks a point or rectangular range in a drawing or bitmap.

Field	Type	Description
type	String	The range type: "rect".
page	Number	The 0-based index of the page containing the range.
section	Number	The 0-based index of the section containing the range (includes hidden pages)
left	Number	The x coordinate of the left edge of the rectangle.
top	Number	The y coordinate of the top edge of the rectangle.
right	Number	The x coordinate of the right edge of the rectangle. For point ranges, this will be equal to left.
bottom	Number	The y coordinate of the bottom edge of the rectangle. For point ranges, this will be equal to top.
units	String	Units for the left, top, right, and bottom properties. Must be one of: "px" (pixels), "tw" (twips). If not specified, the default value of "px" is used.

11.5 OIT.view

The OIT.view functions manage the rectangular area inclusive of toolbars and the displayed document itself, for a document that has been loaded into a DOM element of a host web page.

11.5.1 OIT.view.attach

```
OIT.view.attach( containerElem )
```

This attaches the Web View to an element in a page.

- *containerElem*: A DOM node, typically for a element, that will contain the Web View of files loaded through subsequent calls to OIT.document.load().

HTML File Example

```
<body>
...
<div class="myViewContainer" id="container"></div>
...
</body>
```

Javascript Example

```
var viewContainer = document.getElementById("container");
OIT.view.attach(viewContainer);
...
OIT.view.load("primary_output_file.html");
```

11.5.2 OIT.view.info

```
OIT.view.info()
```

Returns an object containing information about the Web View export process and the converted document. This object contains the following fields:

- *fi* : The integer FI id for the input document. See SCCFI.H in the Web View sdk/common directory for a list of possible FI values.
- *fistring*: The name of the input document's format id
- *product*: The name of the OIT product that generated the output (typically "Outside In® Technology")
- *version*: The version number of the OIT product that generated the output

Return Value

The info object for the currently loaded document

11.5.3 OIT.view.load

```
OIT.view.load( url )
```

This loads a WebView document into an element on the page previously specified by OIT.view.attach().

- *url*: The url of a primary output file produced by Web View Export. This file must have been produced in Ajax mode.

Note: This function uses Ajax methods to load the output of Web View Export into a container on the current page. This function will fail if OIT.view.attach was not called first, to specify the container into which the web view may be loaded.

If the file whose URL is provided was not produced by Web View Export, or produced without the document structure option set to one of the Ajax modes, this function will fail.

11.5.4 OIT.view.unload

```
OIT.view.unload()
```

Unloads a Web View Export document previously loaded via OIT.view.load. It is not necessary to call this function when viewing successive documents in a single view; loading a new document into a view will automatically cause the previous document to be unloaded.

11.5.5 OIT.view.fitToContainer

```
OIT.view.fitToContainer()
```

This function resizes the web view to fill the dimensions of the container to which it has been attached. It is only necessary to call this function when the container has been resized independently of the browser window.

11.5.6 OIT.view.width

```
OIT.view.width( [x] )
```

- x - An optional CSS size value for the new width of the web view. A numeric value will be interpreted as a pixel value.

Return Value

A number representing the width of the current web view in pixels, including any toolbars.

11.5.7 OIT.view.height

```
OIT.view.height( [y] )
```

- y - An optional CSS size value for the new height of the web view. A numeric value will be interpreted as a pixel value.

Return Value

A number representing the height of the current web view in pixels, including any toolbars.

11.6 OIT.document

This section describes the OIT.document class.

11.6.1 OIT.document.properties

```
OIT.document.properties()
```

Returns the document properties object, which contains the document properties from the input document. This object may contain any combination of the following fields:

- **general.title**: The document title
- **general.author**: The document author name
- **general.company**: The document company
- **general.subject**: The document subject
- **general.keywords**: The document keywords
- **general.category**: The document category
- **general.creationDate**: The date the document was created
- **general.lastSaveDate**: The date the document was last saved
- **statistics.backupDate**: The date the document was last backed up
- **statistics.completedDate**: The date the document was completed
- **statistics.charCount**: The number of characters in the document
- **statistics.pageCount**: The number of pages in the document
- **statistics.wordCount**: The number of words in the document
- **statistics.byteCount**: The number of bytes in the document
- **statistics.lineCount**: The number of lines in the document
- **statistics.paraCount**: The number of paragraphs in the document
- **statistics.slideCount**: The number of slides in the document
- **statistics.hiddenSlideCount**: The number of hidden slides in the document
- **statistics.noteCount**: The number of notes in the document
- **statistics.charCountWithSpaces**: The number of characters in the document, including spaces
- **statistics.editMinutes**: The total number of minutes spent editing the document
- **statistics.lastPrintDate**: The date the document was last printed
- **statistics.lastSavedBy**: The person who last saved the document
- **statistics.revisionDate**: The date of the current document revision
- **statistics.revisionNumber**: The current revision number
- **statistics.versionDate**: The date of the document version
- **statistics.versionNumber**: The document version number
- **other**: An array of name/value pairs for any document properties not included in the above fields. Each entry in this array is an object with "name" and "value" properties.

Return Value

The document properties object for the currently loaded document.

11.6.2 OIT.document.externalData

```
OIT.document.externalData()
```

Returns an object containing the external data added to the web view at the time it was generated. These values are set during the export process via the EXAddKeyValue API function.

For example, if the following call was made at the time a file was exported:

```
EXAddKeyValueInt( hExport, (VTLPCSTR)"UserId", 42 );
```

Then the following Javascript statements would be possible in the browser:

```
var data = OIT.document.externalData();
if( data )
  alert( data.UserId );  // displays "42"
```

Return Value

An object containing key/value data applied via the Export API, or undefined if no such data is available.

11.6.3 OIT.document.setOptions

```
OIT.document.setOptions( options )
```

Initializes the viewer with the specified options.

options: An object with zero or more of these fields set:

Field	Description
disableAnimation	A boolean flag controlling whether animations are enabled for various transitions such as scrolling. If true, animations will be disabled. By default, or if disableAnimation is set to false, animations will be enabled.
memoryModel	A string value that affects how much memory is used to store downloaded pages in the browser. When this memory limit is exceeded, page content may be discarded from memory but will be reloaded from the server on demand. Note that this is a soft limit, and may be exceeded under certain circumstances. This option does not have any effect unless the SCCOPT_OUTPUT_STRUCTURE option is set to either SCCWV_STRUCTURE_AJAX_CHUNKED or SCCWV_STRUCTURE_AJAX_STREAMING. May have any of the following values: "none", "large", "small", "micro". By default, set to "large" for desktop browsers and "small" for mobile browsers. "none" removes any limits on memory. "micro" places a very small limit on memory, and is mainly useful for testing purposes.

11.6.4 OIT.document.zoom

```
OIT.document.zoom( [zoom] )
```

Sets or retrieves the zoom factor for the view. Calling this function will cause a 'zoom' event to be dispatched if the zoom factor changes. If zoom is called with no arguments, it simply returns the current zoom factor.

zoom: May be a numeric zoom factor (1.0 = 100%, 1.5 = 150%, etc.) or one of the following strings:

Value	Description
"pagewidth"	Set the zoom factor so the widest page in the document will fill the view from left to right.
"pageheight"	Set the zoom factor so the tallest page in the document will fill the view from top to bottom.
"page"	Set the zoom factor so the largest page in the document will fit within the view.

Return Value

Returns a number indicating the current zoom factor (1.0 = 100%, 1.5 = 150%, etc.)

11.6.5 OIT.document.zoomIn

```
OIT.document.zoomIn()
```

Zooms in one step. Calling this function will cause a 'zoom' event to be dispatched if the zoom factor is not already at the maximum.

Return Value

Returns a number indicating the current zoom factor (1.0 = 100%, 1.5 = 150%, etc.)

11.6.6 OIT.document.zoomOut

```
OIT.document.zoomOut()
```

Zooms out one step. Calling this function will cause a 'zoom' event to be dispatched if the zoom factor is not already at the minimum.

Return Value

Returns a number indicating the current zoom factor (1.0 = 100%, 1.5 = 150%, etc.)

11.6.7 OIT.document.rotate

```
OIT.document.rotate( angle )
```

Rotates all of the pages in the document by the specified angle. If rotate is called with no arguments, it simply returns the current rotation angle. Note that rotation is not supported for all file types. Calling rotate() for a display engine that does not support rotation will have no effect and the angle 0.0 will always be returned.

angle: The angle to rotate the pages, in degrees. Must be an integer multiple of 90 degrees.

Return Value

Returns a number indicating the current rotation angle.

11.6.8 OIT.document.showHidden

```
OIT.document.showHidden( [mode] )
```

For spreadsheet and database files, makes hidden rows and/or columns visible. This affects the current sheet only and is not saved when the current sheet changes. When a different spreadsheet page is displayed, it will behave as if OIT.document.showHidden("none") was called.

- mode - The new visibility mode. Must be one of the following:

  all - Makes hidden and narrow rows and columns visible by increasing their width to a preset minimum.

  none - Resets all rows and columns to their default height/width. Hidden rows and columns are set to 0 height/width.

  rows - Makes hidden and narrow rows visible by setting them to a minimum height. Columns are not affected.

  columns - Makes hidden and narrow columns visible by setting them to a minimum width. Rows are not affected.

11.6.9 OIT.document.find

```
OIT.document.find( searchTerm, callbackFn, [caseSensitive], [direction] [startacc]
)
```

Performs an asynchronous search for the specified text string in the specified direction, starting at the current position.

searchTerm: A string containing the search term to look for.

callbackFn: This function will be called when the search term is found. A TextRange is passed as the only parameter to the callback function.

If the search term is not found anywhere in the specified direction, the callback function will be called with no arguments.

caseSensitive: A boolean value specifying whether searches are case sensitive. May be true (the default) or false.

direction: A string value specifying the search direction from the current text position. May be "down" (the default) or "up".

startacc: The starting ACC (actual character count) for the search operation. If not specified, defaults to the end of the document if direction is "up" or the beginning of the document otherwise.

To continue searching from the location of the previous OIT.document.find result, pass in (range.acc+1) for startacc (or (range.acc-1) if direction is 'up').

11.6.10 OIT.document.findAll

```
OIT.document.findAll( searchTerm, callbackFn, [caseSensitive] )
```

Performs an asynchronous document-wide search for the specified text string.

searchTerm: A string containing the search term to look for.

callbackFn: This function will be called each time the search term is found. A TextRange is passed as the only parameter to the callback function. If the callback function returns false, the search operation will be aborted.

If the search term is not found anywhere in the specified direction, the callback function will be called a single time with no arguments.

caseSensitive: A boolean value specifying whether searches are case sensitive. May be true (the default) or false.

11.6.11 OIT.document.getRawText

```
OIT.document.getRawText( callbackFunction )
```

callbackFunction: A Javascript function provided by the caller that will be called asynchronously by the Web View Javascript library. The function must conform to the following prototype:

```
function rawTextCallback( buffer, acc )
```

- buffer: Javascript string containing utf8-encoded Unicode text from the document
- acc: The acc of the first Unicode character in the text buffer. The acc of the last character in the buffer is equal to (acc + buffer.length - 1).

This function will be called repeatedly until all of the text of the file has been provided, unless the rawTextCallback function returns a boolean value of false, in which case the text retrieval will stop and no further calls will be made to the callback function. When the end of the document text is reached, the callback function will be called a single time with no arguments.

11.6.12 OIT.document.highlight

```
OIT.document.highlight( searchTerm, style, [callbackFn], [caseSensitive],
[properties] )
```

Performs an asynchronous document-wide search for the specified text string and adds a highlight anywhere it occurs.

searchTerm: A string containing the search term to look for.

style: The style to apply to each highlight. A CSS string that may include the color and/or background-color properties. Any other properties are ignored.

callbackFn: This (optional) function will be called each time the search term is found. The function will be called with two arguments: a TextRange and a highlight id. If the callback function returns false, the search operation will be aborted.

If the search term is not found anywhere in the specified direction, the callback function will be called a single time with no arguments.

caseSensitive: A boolean value specifying whether searches are case sensitive. May be true (the default) or false.

properties: A Javascript object that contains additional data for the highlight. This object will be available to event handlers when events occur on this object, and may contain private data for the application's use.

11.6.13 OIT.document.select

```
OIT.document.select( range, moveto )
```

Makes the specified text range, cell range, or rect range the active selection. If range is null, any selection is cleared. If moveto is true, the display will be moved so the selection is visible.

range: Specifies the range to select. Must be a valid OIT TextRange, CellRange or RectRange, or a DOM Range (from document.createRange() and other methods), or a DOM Selection (from window.getSelection() and other methods).

moveto: If false, the range is selected without any other effect. Otherwise, the range is made visible just as if OIT.document.moveto were called.

Return Value

True if the selection changed, false if it was not changed.

11.6.14 OIT.document.selectionType

```
OIT.document.selectionType()
```

Retrieves the current selection type.

Return Value

"text", "cell", "rect" or "none".

11.6.15 OIT.document.selectionMode

```
OIT.document.selectionMode( [mode] )
```

Queries and optionally sets the selection mode.

mode: An optional parameter specifying the selection mode. A string value of "text", "rect", "cell", or "none". Only modes supported by the current view will be accepted. Attempting to set the selection mode to an unsupported value (i.e., "text" when viewing a spreadsheet) will have no effect. Setting the selection mode to "none" will disable selection.

Return Value

The current selection mode ("text", "rect", "cell", or "none").

11.6.16 OIT.document.selectionStyle

```
OIT.document.selectionStyle( [style] )
```

Queries and optionally sets the selection style. This determines the appearance of the selection rectangle.

style: An optional CSS string parameter specifying the selection style. Passing in null for this value will reset the selection style to the default (which is determined by the oit-sel-rect CSS class).

Return Value

The current selection style, or null if the default is being used.

11.6.17 OIT.document.selection

```
OIT.document.selection()
```

Returns a range representing currently selected text, rect, or cell range.

Return Value

A TextRange, RectRange, or CellRange object, or undefined if nothing is selected.

11.6.18 OIT.document.textSelection

```
OIT.document.textSelection()
```

Returns a range representing currently selected text.

Return Value

A TextRange object, or undefined if no text is selected.

11.6.19 OIT.document.cellSelection

```
OIT.document.cellSelection()
```

Returns an object representing the currently selected range of cells.

Return Value

A CellRange object, or undefined if no cells are selected.

11.6.20 OIT.document.rectSelection

```
OIT.document.rectSelection()
```

Returns an object representing the currently selected rectangle or point.

Return Value

A RectRange object, or undefined if no rectangular range is selected.

11.6.21 OIT.document.textRange

```
OIT.document.textRange( domRange )
```

Returns an object representing the range of text specified by a DOM Range object (from document.createRange() and other methods). If the passed in Range object does not begin and end in the document text, will return undefined.

Return Value

A TextRange object.

11.6.22 OIT.document.textRange

```
OIT.document.textRange( domSelection )
```

Returns an object representing the range of text specified by a DOM Selection object (from window.getSelection() and other methods). If the passed in Selection object does not begin and end in the document text, will return undefined.

Return Value

A TextRange object.

11.6.23 OIT.document.textRange

```
OIT.document.textRange( acc, accend )
```

Returns an object representing the range of text specified by the acc and accend values

acc: The starting ACC for the text range.

accend: The ending ACC for the text range.

Return Value

A TextRange object.

11.6.24 OIT.document.cellRange

```
OIT.document.cellRange( [sheet], c1, r1, [c2], [r2] )
```

Returns an object representing the cell or range of cells at the specified spreadsheet or database table, row(s), and column(s).

sheet: The sheet name or 0-based index. If unspecified, the current sheet will be used.

c1: The 0-based index of the starting column for the selection.

r1: The 0-based index of the starting row for the selection.

c2: The 0-based index of the ending column for the selection. If unspecified, the ending column will be the same as the starting column.

r2: The 0-based index of the ending row for the selection. If unspecified, the ending row will be the same as the starting row.

Return Value

A CellRange object.

11.6.25 OIT.document.rectRange

```
OIT.document.rectRange( [page], x1, y1, [x2], [y2], [units] )
```

Returns an object representing an x,y position or rectangle on the specified page.

page: The page name or 0-based index. If unspecified, the current page will be used.

left: The x coordinate for the left edge of the rectangle.

top: The y coordinate for the top edge of the rectangle.

right: The x coordinate for the right edge of the rectangle. If unspecified, the right edge will be equal to the left edge.

bottom: The y coordinate for the bottom edge of the rectangle. If unspecified, the bottom edge will be equal to the top edge.

units: Units for the left, top, right, and bottom properties. Must be one of: "px" (pixels), "tw" (twips). If not specified, the default value of "px" is used.

Return Value

A RectRange object.

11.6.26 OIT.document.updateViewSize

```
OIT.document.updateViewSize()
```

The Web View Export view periodically updates it's layout in response to changes in the size of the view div, but there may be a slight delay while this update is processed. To update the display immediately after a change in view size, OIT.document.updateViewSize() may be called.

11.6.27 OIT.document.addEventListener

```
OIT.document.addEventListener( name, fn )
```

Adds an event listener for OIT document events. The event listener function receives the event as it's argument. event.preventDefault() may be called to prevent the default behavior described below.

See Events from the Javascript API for information on how events are handled.

Name may be any of the following:

Event Name	Description	Event Detail	Default Behavior
load	Similar to the DOM load event, this event is fired when the document has been loaded and the Web View Export viewer initialized.	**url**: The url of the document that has been loaded.	(none)
zoom	This event is fired when the user changes the zoom level in the Web View Export display.	**zoom**: A number representing the current scaling level. 1.0 = no zoom, 1.5 = image is 1.5 times normal size, etc. **mode**: An optional string stating the zoom mode. Will be "page", "pagewidth" or undefined.	(none)
rotate	This event is fired when the user changes the rotation angle in the Web View Export display.	**angle**: A number representing the current rotation angle in degrees: 0, 90, 180, or 270.	(none)
idle	This event is fired periodically when no user mouse or keyboard activity is detected.	(none)	(none)
error	This event is fired when an error occurs in the client side OIT Javascript code.	**message**: A human readable message describing the error.	Log the error to the Javascript console.
select	This event is fired when the document selection changes.	**selection**: The new selection (may be undefined if nothing is selected). **empty**: A boolean value indicating whether this is an empty selection. An empty text or rect selection occurs when the user clicks on the document but does not drag to create a selection. Empty cell selections are not possible since at least one cell will always be selected. **highlightid**: If the selection is changing due to a highlight being activated, this will contain the highlight id.	Set the current highlight to the one closest to the new selection.

Event Name	Description	Event Detail	Default Behavior
scroll	This event is fired when the view is scrolled. It is also fired when the scroll range changes, in response to a page change, zoom, or rotate operation and when the document is first loaded.	**viewUpperLeft.pageIndex**: The 0-based index of the page closest to the upper left corner of the view **viewUpperLeft.x, viewUpperLeft.y**: The coordinates of the upper left corner of the view relative to the page specified by viewUpperLeft.pageIndex **viewLowerRight.pageIndex**: The 0-based index of the page closest to the bottom right corner of the view **viewLowerRight.x, viewLowerRight.y**: The coordinates of the lower right corner of the view relative to the page specified by viewLowerRight.pageIndex Note that the x and y coordinates may fall outside the page. The OIT.pages.dimensions() function may be called to retrieve the dimensions of each page.	(none)
keydown	This event is fired when a keydown event is received by the view.	**srcEvent**: The original KeyboardEvent that triggered this event	Various actions, depending on the pressed key

11.6.28 OIT.document.removeEventListener

```
OIT.document.removeEventListener( name, fn )
```

Removes an OIT document event listener that was previously added by a call to OIT.document.addEventListener.

See Events from the Javascript API for information on how events are handled.

11.7 OIT.pages

This section describes OIT.pages.

11.7.1 OIT.pages.current

```
OIT.pages.current()
```

Returns the index of the page currently nearest the top of the display.

Return Value

A number representing the 0-based index of the current page.

11.7.2 OIT.pages.pageName

```
OIT.pages.pageName( [index] )
```

Returns the name of the specified page.

index: The 0-based index of a page in the range [0..(OIT.pages.count()-1)]. If not specified, returns the name of the current page.

Return Value

A string containing the name of the specified page.

11.7.3 OIT.pages.count

```
OIT.pages.count()
```

Returns the number of pages in the document.

Return Value

The number of pages in the document.

11.7.4 OIT.pages.moveto

```
OIT.pages.moveto( dest )
```

Moves the view to the specified destination. Calling this function will cause a 'change' event to be dispatched if the current page changes.

dest: The destination to move to. May be a 0-based page index, an element or text node in the body of the document, or a text range, cell range, or rect range object.

Return Value

A number representing the 0-based index of the current page.

11.7.5 OIT.pages.movetoNext

```
OIT.pages.movetoNext()
```

Moves the view to the next page in the document. Calling this function will cause a 'change' event to be dispatched if the view is not already at the end of the document.

Return Value

A number representing the 0-based index of the current page.

11.7.6 OIT.pages.movetoPrev

```
OIT.pages.movetoPrev()
```

Moves the view to the previous page in the document. Calling this function will cause a 'change' event to be dispatched if the view is not already at the beginning of the document.

Return Value

A number representing the 0-based index of the current page.

11.7.7 OIT.pages.info

```
OIT.pages.info( [pageIndex] )
```

Retrieves information about the specified page.

pageIndex: The 0-based index of the page to retrieve dimensions for. If not specified, the dimensions of the current page will be returned.

Return Value

An object with the following properties:

- width: The width of the page, in pixels, at 100% zoom and 0 degrees rotation. Changing the zoom level or page rotation angle will not affect this value.
- height: The height of the page, in pixels, at 100% zoom and 0 degrees rotation. Changing the zoom level or page rotation angle will not affect this value.
- type: The type of this page, as defined in its original document (see below).

The page type will be one of the following:

- "page" - Word processor page
- "worksheet" - Spreadsheet worksheet
- "database" - Database table
- "bitmap" - Bitmap image
- "archive" - File archive
- "drawing" - Vector drawing
- "slide" - Presentation slide
- "html" - HTML page
- "chart" - Spreadsheet chart
- "emailarchive" - Email archive
- "email" - Email document

11.7.8 OIT.pages.addEventListener

```
OIT.pages.addEventListener( name, fn )
```

Adds an event listener for OIT page events. The event listener function receives the event as its argument. event.preventDefault() may be called to prevent the default behavior described below.

See Events from the Javascript API for information on how events are handled.

Name may be any of the following:

Event Name	Description	Event Detail	Default Behavior
change	Called whenever the currently displayed page (that is, the value returned by OIT.pages.current()) changes	**page**: The index of the page being displayed. **features**: The features supported for this page type. An object with the following fields: "zoom", "pages", "highlights". **width**: The width of the page, in pixels, at 100% zoom and 0 degrees rotation. Changing the zoom level or page rotation angle will not affect this value. **height**: The height of the page, in pixels, at 100% zoom and 0 degrees rotation. Changing the zoom level or page rotation angle will not affect this value. **type**: The page type. See OIT.pages.info for a list of page types.	(none)
add	Called when information on additional pages has been downloaded from the server. This message will only be dispatched when the file was exported using streaming mode.	**startPage**: The index of the first added page	(none)

Example:

```
OIT.pages.addEventListener( "change", function(event)
{
document.getElementById("pagenum").innerHTML = event.detail.page;
});
```

11.7.9 OIT.pages.removeEventListener

```
OIT.pages.removeEventListener( name, fn )
```

Removes an OIT page event listener that was previously added by a call to OIT.pages.addEventListener.

See Events from the Javascript API for information on how events are handled.

11.8 OIT.highlights

This section describes management of highlights.

11.8.1 Adding and Removing Highlights

This section describes adding and removing highlights.

11.8.1.1 OIT.highlights.add

```
OIT.highlights.add( style, [range], [properties], [comment] )
```

This method adds a highlight to a range of the document, using the specified style.

- *style*: A CSS string that may include the color and/or background-color properties. Any other properties are ignored.
- *range*: May be a text range (from OIT.document.textRange() or OIT.document.textSelection()), a cell range (from OIT.document.cellRange() or OIT.document.cellSelection()) or a rect range (from OIT.document.rectRange() or OIT.document.rectSelection()). When no range parameter is specified, the current selection is used. If there is no current selection, the function has no effect.
- *properties*: An optional Javascript object that contains additional data for the highlight. This object will be available to event handlers when events occur on this object, and may contain private data for the application's use. Highlight properties may be changed or retrieved at any time by calling OIT.highlights.properties() or OIT.highlights.property().
- *comment*: An optional comment string for the highlight.

Return Value

This method returns a numeric id, which can be used to later identify this particular highlight for other actions in the API, such as adding a comment.

Examples:

```
var hlProps= {user:"bobT",
                date:"05-29-2013 12:14 PST",
                userGroup: "administrators" };

OIT.highlights.add( "background-color:#FFFF00",  hlProps, "This is a comment " );
```

11.8.1.2 OIT.highlights.defineStamp

```
OIT.highlights.defineStamp( name, imageUrl, [options] )
```

This method specifies an image that may be used for image-stamp highlighting. Only images that have been named through this method may be used as for highlighting.

- *name*: This is a name that will be used to reference the image stamp.
- *imageUrl*: This is the URL to a PNG, JPG, or GIF file to be used as an image stamp
- *options*: An optional object, which if present may contain properties that affect the formatting of the image stamp. Allowable properties are listed below.

Stamp Formatting Options

Besides the image itself, there are additional options that can affect the display of an image-stamp highlight. These may be specified as properties in an options object supplied to OIT.highlights.defineStamp, or as properties of a stamp object supplied to OIT.highlights.addStamp. Supported options include the following:

- *sizing*: Indicates how stamp images should be sized. Possible values are:

  "scale" - The stamp will be fit into the destination rectangle at the largest size that preserves its aspect ratio.

  "stretch" - The stamp will be stretched to entirely fill the destination rectangle, with no preservation of aspect ratio.

"image" - The stamp will be applied at the native size of the image, regardless of the boundaries of the rectangle.

Default: "scale"

The top left corner of the stamp will always align with the top left corner of the destination rectangle.

- *opacity*: A value from 0 to 1.0 indicating the opacity of the image. A value of 0 indicates complete transparency; a value of 1 indicates complete opacity.

  Default: 1

- *defaultSize*: An object that contains two properties, x and y; these numbers specify, respectively, a default width and height in pixels for stamps applied at a point (this includes stamps applied via OIT.highlights.autoHighlight).

  Default: none (When not specified, the size of the stamp image is used as the default size.)

  Example:

```
options.defaultSize = {x:160,y:200};
```

Note: None of these values are required to be specified; default values will be used if they are not. If these options are supplied both in OIT.highlights.defineStamp and OIT.highlights.addStamp, the values supplied in addStamp will take precedence.

11.8.1.3 OIT.highlights.addStamp

```
OIT.highlights.addStamp( stamp, [range], [properties], [comment] )
```

This method adds a highlight to a range of the document, using the specified style.

- *stamp*: Specifies the stamp to be applied. This parameter may be either:
  - A string, in which case it specifies the name of a stamp previously specified in OIT.highlights.defineStamp. or
  - An object, in which case it must contain at least a property named "name" which specifies the name of a previously defined stamp. This object may also include one or more of the properties described above as stamp formatting options.
- *range*: may be a rectangular range (from OIT.document.rectRange() or OIT.document.rectSelection()), or a cell range (from OIT.document.cellRange() or OIT.document.cellSelection()).

  In the special case of a rectangular range whose top-left and bottom-right corners are the same, indicating a single point, the image stamp will be applied at its native width and height, centered around the specified point.

  When no range parameter is specified, the current selection is used. If there is no current selection, the function has no effect.

- *properties*: A Javascript object that contains additional data for the highlight. This object will be available to event handlers when events occur on this object, and may contain private data for the application's use.

Return Value

This method returns a numeric id, which can be used to later identify this particular highlight for other actions in the API, such as adding a comment.

Examples

```
OIT.highlights.defineStamp( "approved", "/images/greencheck.png" );
OIT.highlights.defineStamp( "rejected", "/images/redx.png", {sizing:image} );

OIT.highlights.addStamp( "approved", OIT.document.rectRange( 0,0,100,100) );
OIT.highlights.addStamp( { name:"rejected", opacity: .5 } );
```

11.8.1.4 OIT.highlights.apply

```
OIT.highlights.apply( highlights, filterFunction )
```

This method allows an application to specify an entire set of pre-defined highlights.

highlights: This is a JSON-encoded list of highlights previously obtained via a call to the OIT API.

filterFunction: [Optional] A function that can be used to indicate whether or not a particular highlight should be applied.

Notes: If present, the filter function will be called for every highlight, and should return a boolean value. If the function returns true, the highlight in question will be applied to the current document; if the function returns false, the highlight in question will not be applied.

The filter function has the following prototype:

```
function filterFunction( id, properties )
```

The first parameter is the id value that will be assigned to the highlight if the filter function returns true. This is equivalent to an id value returned by OIT.highlights.add(), with the exception that this id value is not yet valid, and may not be used in API calls within the scope of the filter function. The second parameter will be the properties object, if any, that has been defined for the highlight.

Example:

```
OIT.highlights.apply( function(id, props) {
   if( props && props.userId == myData.currentUserId )
     return true;
   else
     return false;
  });
```

11.8.1.5 OIT.highlights.remove

```
OIT.highlights.remove( [id] )
```

Removes the highlight with the specified id. If id isn't specified, removes the current highlight.

11.8.1.6 OIT.highlights.clear

```
OIT.highlights.clear( [filterFunction] )
```

Removes all highlights from the document.

filterFunction [Optional]: A function that can be used to indicate whether or not a particular highlight should be applied.

Note: If present, the filter function will be called for every highlight, and should return a boolean value. If the function returns true, the highlight in question will be removed from current document; if the function returns false, the highlight in question will not be removed.

The filter function has the following prototype:

```
function filterFunction( id, properties )
```

The first parameter is the id of the highlight, as returned by OIT.highlights.add(). The second parameter is the properties object, if any, that has been defined for the highlight.

Example:

```
OIT.highlights.clear( function(id, props) {
   if( props && props.name == "remove me" )
     return true;
   else
     return false;
  });
```

11.8.1.7 OIT.highlights.autoHighlight

```
OIT.highlights.autoHighlight( type, options )
```

This method enables and disables automatic addition of highlights to a web view in response to user selections.

- *type*: One of the following values:
  - "selection" (string value) - Selections will be automatically highlighted
  - "stamp" (string value) - Image stamp highlights will be automatically applied where the user clicks or taps on the document
  - false (boolean value) - Auto highlighting will be turned off. If this parameter is omitted, the method returns the current type of auto-highlighting.
- *options*: An object that specifies parameters for auto-highlighting. Possible properties of this object include:
  - comment [optional] - This property may be a string or boolean value. If it is string, its contents will be applied as a comment to each highlight; if it is the boolean value true, the user will be prompted to enter a comment for each new highlight.

    default value: false
  - sticky [optional] - A boolean value. If present and equal to true, autoHighlight will remain enabled until disabled through another call to this function

    default value: false
  - style [required if type is "selection"; ignored otherwise] - See description of the style parameter to OIT.highlights.add.
  - stamp [required if type is "stamp"; ignored otherwise] - See description of the stamp parameter to OIT.highlights.addStamp.

– properties [optional] - See description of "properties" parameter to OIT.highlights.add; if specified, these properties will be applied automatically to each highlight

  default value: none

Return Value

This method returns the current type of auto-highlighting.

Examples

```
OIT.highlights.autoHighlight( "selection", {style:"background-color:#FFFF00"});
```

11.8.1.8 OIT.highlights.autoHighlightOptions

```
OIT.highlights.autoHighlightOptions([options] )
```

This method sets or updates the current highlight options used by OIT.highlights.autoHighlight.

options: an object that specifies any of the properties associated with auto-highlighting. Any other properties are ignored. This parameter may be omitted if the caller simply wants to query the current value of the auto-highlight options.

Return Value

This method returns an object containing the active values of the auto-highlight options.

11.8.1.9 OIT.highlights.serialize

```
OIT.highlights.serialize( filterFunction )
```

Serializes the highlights in this document so that they may be saved and re-applied later.

filterFunction: [Optional] A function that can be used to indicate whether or not a particular highlight should be included in the serialized data.

Return Value

A Javascript String object that contains all of the highlight and comment information for the current document. This data may be saved and re-applied later via a call to OIT.highlights.apply. If *changesOnly* is set to **true**, this data will only include highlights created by the user since this document was loaded in the browser.

Notes: If present, the filter function will be called for every highlight, and should return a boolean value. If the function returns true, the highlight in question will be serialized; if the function returns false, the highlight in question will not be included in the serialized data.

The filter function has the following prototype:

```
function filterFunction( id, properties )
```

The first parameter is the id of the highlight in question. This is the same value returned by OIT.highlights.add(). The second parameter will be the properties object, if any, that has been defined for the highlight.

Example:

```
function checkProps(id, props)
{
```

```
   if( props && props.userId == myData.currentUserId )
     return true;
   else
     return false;
}

OIT.highlights.serialize( checkProps );
```

11.8.2 Interacting with Highlights

This section describes interacting with highlights.

11.8.2.1 OIT.highlight.setOptions

```
OIT.highlights.setOptions( id, options )
```

Sets options that control how a particular highlight behaves.

id: The highlight id.

options: An object with zero or more of these fields set:

- enableUserDelete: A boolean value controlling whether the user is able to delete the highlight by pressing the DELETE key while the highlight is active. By default, highlights may be deleted by the user.
- enableUserMove: A boolean value controlling whether the user is able to move the highlight using the mouse, touch, or keyboard. This applies to rectangular and cell highlights only. By default, highlights may be moved by the user.
- enableUserComment: A boolean value controlling whether the user is able to add, edit, or remove the highlight comment. By default, comments are enabled.

11.8.2.2 OIT.highlights.comment

```
OIT.highlights.comment( id, [text] )
```

Optionally, sets the comment text for the specified highlight, and returns the highlight comment.

id: The highlight id.

text: The comment text string. If the highlight has an existing comment associated with it, the text is replaced by the new comment text. If a value is not passed in for this parameter, the current comment will be returned unchanged. Passing in null for this parameter will remove the comment from the highlight.

Return Value

String containing the comment text, or undefined if an invalid highlight id was specified or the highlight has no comment.

11.8.2.3 OIT.highlights.properties

```
OIT.highlights.properties( id, [properties] )
```

Optionally, sets the custom properties for the specified highlight, and returns the highlight custom properties.

id: The highlight id.

properties: An optional javascript object that contains additional data for the highlight. This object replaces any existing properties on the highlight. This object will be

available to event handlers when events occur on this object, and may contain private data for the application's use.

Return Value

The properties object attached to the highlight, or undefined if the highlight has no properties or if the specified id is not a valid highlight id.

11.8.2.4 OIT.highlights.property

```
OIT.highlights.property( id, valueName, [value] )
```

Optionally, sets a single custom property for the specified highlight, and returns the value of the property. If the specified highlight has a properties object, the named value will be added to it. If the named property already existed in the properties object, it will be replaced. If the specified highlight does not have a properties object associated with it, one will be created with the supplied property value.

id: The highlight id.

valueName: The name string for the property to be set.

value: The optional value of the property to be set. This parameter may be of any type. This value replaces any existing value for the specified property. If a value is not provided for this parameter, the value of the property is returned unchanged.

Return Value

The value of the specified property

11.8.2.5 OIT.highlights.current

```
OIT.highlights.current()
```

Returns the id of the current highlight. The current highlight is the one closest to the most recent selection or click. If there is no current highlight, returns undefined.

11.8.2.6 OIT.highlights.active

```
OIT.highlights.active()
```

Returns the id of the active highlight. The active highlight is the one that was most recently clicked or selected. If there is no active highlight, returns undefined.

11.8.2.7 OIT.highlights.moveto

```
OIT.highlights.moveto( [id] )
```

Moves the view to the highlight with the specified id. If id isn't specified, moves to the current highlight.

11.8.2.8 OIT.highlights.movetoNext

```
OIT.highlights.movetoNext()
```

Moves the view to the next highlight in document order.

11.8.2.9 OIT.highlights.movetoPrev

```
OIT.highlights.movetoPrev()
```

Moves the view to the previous highlight in document order.

11.8.2.10 OIT.highlights.range

```
OIT.highlights.range( [id] )
```

Returns the text range, cell range, or rect range associated with a highlight. If id isn't specified, returns the range of the current highlight.

11.8.2.11 OIT.highlights.showComment

```
OIT.highlights.showComment( [id], [options] )
```

Makes the comment associated with a highlight visible. If id isn't specified, shows the comments for the current highlight. If the specified highlight has no comment, this function has no effect.

options: An object with fields controlling the display of the comment. The object may have any combination of the following fields:

Field	Description
mode	A string value controlling whether the comment is editable. May be one of the following: ■ **new**: Used when adding a new comment. If the user cancels the dialog, the highlight is automatically removed. ■ **edit**: Entire comment is editable, including the existing value. ■ **view**: Comment is not editable (the default). Note that the new and edit modes will allow the comment to be added, edited or removed regardless of the value of the enableUserComment option.

11.8.2.12 OIT.highlights.hideComment

```
OIT.highlights.hideComment( [id] )
```

Makes the comment associated with a highlight invisible. If id isn't specified, hides the comments for the current highlight. If the specified highlight has no comment, this function has no effect.

11.8.2.13 OIT.highlights.first

```
OIT.highlights.first()
```

Returns the id of the first highlight in the document.

11.8.2.14 OIT.highlights.last

```
OIT.highlights.last()
```

Returns the id of the last highlight in the document.

11.8.2.15 OIT.highlights.next

```
OIT.highlights.next( [id] )
```

Returns the id of the next highlight in the document after the specified id. If id isn't specified, returns the id of the next id after the current highlight.

11.8.2.16 OIT.highlights.prev

```
OIT.highlights.prev( [id] )
```

Returns the id of the previous highlight in the document from the current position. If id isn't specified, returns the id of the previous id before the current highlight.

11.8.2.17 OIT.highlights.activate

```
OIT.highlights.activate( [id] )
```

Activates the highlight with the specified id. If id isn't specified, activates the current highlight.

11.8.2.18 OIT.highlights.addEventListener

```
OIT.highlights.addEventListener( name, fn )
```

Adds an event listener for OIT highlight events. The event listener function receives the event as it's argument. event.preventDefault() may be called to prevent the default behavior described below.

See Events from the Javascript API for information on how events are handled.

Name may be any of the following:

Event Name	Description	Event Detail	Default Behavior
add	Fired when a new highlight has been added.	**highlightid**: The id of the new highlight.	(none)
mode	Fired when the current mode of highlighting changes	**autohighlight**: true if auto-highlighting mode is enabled, false if not. **autocomment**: true if auto-commenting mode is enabled, false if not. **autostyle**: the current style defined for auto-highlighting. **selectionmode**: the current selection mode (text/rect/cell).	(none)
activate	Fired when the current highlight changes.	**highlightid**: The id of the current highlight. **before**: A boolean value indicating if the selection falls before the highlight. **after**: A boolean value indicating if the selection falls after the highlight.	(none)

Event Name	Description	Event Detail	Default Behavior
mousedown	Fired when the user depresses a mouse button over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseup	Fired when the user releases a mouse button over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
click	Fired when the user clicks on a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
dblclick	Fired when the user double clicks on a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
contextmenu	Fired when the user right clicks on a highlight, or takes another action that would normally bring up a context menu.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mousemove	Fired when the user moves the mouse cursor over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseover	Fired when the user moves the mouse cursor over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseout	Fired when the user moves the mouse cursor away from a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original MouseEvent that triggered this event.	(none)
touchstart	Fired when the user places a touch point over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchend	Fired when the user removes a touch point over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)

Event Name	Description	Event Detail	Default Behavior
touchmove	Fired when the user moves a touch point over a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchenter	Fired when a touch point enters a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchleave	Fired when a touch point exits a highlight.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchcancel	Fired when the touch has been disrupted in some way.	**highlightid**: The id of the affected highlight. **srcEvent**: The original TouchEvent that triggered this event.	(none)

11.8.2.19 OIT.highlights.removeEventListener

```
OIT.highlights.removeEventListener( name, fn )
```

Unregisters an event handler that was previously added by a call to OIT.highlights.addEventListener.

See Events from the Javascript API for information on how events are handled.

11.8.2.20 Examples

Searching for text and applying a highlight:

```
var textRange = OIT.document.find("searchtext");
if( textRange )
  OIT.highlights.add( "{background-color:yellow}", textRange );
```

11.9 OIT.hyperlinks

This section describes OIT.highlights.

11.9.1 Interacting with Hyperlinks

This section describes interacting with highlights.

11.9.1.1 OIT.hyperlinks.addEventListener

```
OIT.hyperlinks.addEventListener( name, fn )
```

Adds an event listener for OIT hyperlink events. The event listener function receives the event as it's argument. event.preventDefault() may be called to prevent the default behavior described below.

See Events from the Javascript API for information on how events are handled.

Name may be any of the following:

Event Name	Description	Event Detail	Default Behavior
mousedown	Fired when the user depresses a mouse button over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseup	Fired when the user releases a mouse button over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	(none)
click	Fired when the user clicks on a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **target**: For links to bookmarks in the current document, the target element. **srcEvent**: The original MouseEvent that triggered this event.	Set window.location to the target url (for external hyperlinks) or call OIT.pages.moveto (for internal hyperlinks).

Event Name	Description	Event Detail	Default Behavior
dblclick	Fired when the user double clicks on a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **target**: For links to bookmarks in the current document, the target element. **srcEvent**: The original MouseEvent that triggered this event.	(none)
contextmenu	Fired when the user right clicks on a highlight, or takes another action that would normally bring up a context menu.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mousemove	Fired when the user moves the mouse cursor over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	(none)

Event Name	Description	Event Detail	Default Behavior
mouseover	Fired when the user moves the mouse cursor over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	Set the status bar text to the hyperlink url.
mouseout	Fired when the user moves the mouse cursor away from a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original MouseEvent that triggered this event.	Remove the hyperlink url from the status bar (see mouseover, above).
touchstart	Fired when the user places a touch point over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)

Event Name	Description	Event Detail	Default Behavior
touchend	Fired when the user removes a touch point over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchmove	Fired when the user moves a touch point over a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchenter	Fired when a touch point enters a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)

Event Name	Description	Event Detail	Default Behavior
touchleave	Fired when a touch point exits a highlight.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchcancel	Fired when the touch has been disrupted in some way.	**url**: A string containing the url for the hyperlink that was clicked. **external**: A boolean value specifying whether this is a hyperlink to another document (true = external, false = link to a bookmark in the current document). **range**: A TextRange or RectRange specifying the range affected by the hyperlink, if available. **srcEvent**: The original TouchEvent that triggered this event.	(none)

11.9.1.2 OIT.hyperlinks.removeEventListener

```
OIT.hyperlinks.removeEventListener( name, fn )
```

Unregisters an event handler that was previously added by a call to OIT.hyperlinks.addEventListener.

See Events from the Javascript API for information on how events are handled.

11.10 OIT.archive

This section describes OIT.archive.

11.10.1 Interacting with archive nodes

11.10.1.1 OIT.archive.addEventListener

```
OIT.archive.addEventListener( name, fn )
```

Adds an event listener for OIT archive node events. The event listener function receives the event as it's argument. event.preventDefault() may be called to prevent the default behavior described below.

See Events from the Javascript API for information on how events are handled.

Name may be any of the following:

Event Name	Description	Event Detail	Default Behavior
mousedown	Fired when the user depresses a mouse button over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseup	Fired when the user releases a mouse button over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
click	Fired when the user clicks on an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
dblclick	Fired when the user double clicks on an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
contextmenu	Fired when the user right clicks on an archive node, or takes another action that would normally bring up a context menu.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mousemove	Fired when the user moves the mouse cursor over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseover	Fired when the user moves the mouse cursor over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
mouseout	Fired when the user moves the mouse cursor away from an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original MouseEvent that triggered this event.	(none)
touchstart	Fired when the user places a touch point over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)

Event Name	Description	Event Detail	Default Behavior
touchend	Fired when the user removes a touch point over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchmove	Fired when the user moves a touch point over an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchenter	Fired when a touch point enters an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchleave	Fired when a touch point exits an archive node.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)
touchcancel	Fired when the touch has been disrupted in some way.	**nodeid**: The id of the affected node. **srcEvent**: The original TouchEvent that triggered this event.	(none)

11.10.1.2 OIT.archive.removeEventListener

```
OIT.archive.removeEventListener( name, fn )
```

Unregisters an event handler that was previously added by a call to OIT.archive.addEventListener.

See Events from the Javascript API for information on how events are handled.

11.11 OIT.toolbars

The API supports adding a toolbar to each of the four sides of a web view. The API functions address a toolbar by the edge of the view on which it is located – "top," "bottom," "left," or "right." Toolbars may also be placed on top of the web view, as an overlay toolbar. (This should be used carefully, because while it is visible, an overlay toolbar at least partially obstructs the view of the document.)

In cases where two toolbars might overlap, priority is given to the first toolbar added. For example, if a toolbar is added to the top of the web view, and then another is added to the left side of the web view; the top toolbar will extend across the entire width of the web view, but the toolbar on the left edge will extend from the bottom of the top toolbar to the bottom of the web view.

11.11.1 OIT.toolbars.add

```
OIT.toolbars.add( edge, element, [options] )
```

Adds a toolbar to the Web View Export display.

edge: Must be one of "top," "bottom," "left," or "right." This parameter indicates the edge of the display where the toolbar will be located.

element: This is a DOM element supplied by the caller that contains the toolbar elements.

options: The options object may be used to specify additional properties of the toolbar.

Field	Description
size	The size, in pixels, of the toolbar. ■ If the specified toolbar is attached to the top or bottom of the view, the size value represents the height of the toolbar. ■ If the toolbar is attached to the left or right of the view, the size value represents the width of the toolbar. ■ If a size is not specified, the element supplied for the toolbar must have a size defined for it through CSS.
visible	If true, the toolbar will be shown immediately upon creation. If this parameter is false or not present, the toolbar will be initially hidden.
border	An optional CSS border value to be applied between the toolbar and the document view. For example, a typical value might be "1px solid #c0c0c0". If the toolbar was attached to the top edge of the view, this value will be used as the toolbar's bottom border; if the toolbar was attached to the bottom edge, this value would be used as its top border; etc.
overlay	If true, the toolbar will be placed over the displayed document. (The default behavior is to place toolbars adjacent to the displayed document.)

Return Value

Returns the id of the newly added toolbar.

Example

This is an example of a simple status bar that shows the location of the current document:

```
OIT.document.addEventListener( "load", function(evt) {
 var tools = document.createElement("div");
 tools.appendChild( document.createTextNode(evt.detail.url) );
 OIT.toolbars.add("bottom", tools, {size:20, visible: true} );
}
```

11.11.2 OIT.toolbars.remove

```
OIT.toolbars.remove( id )
```

Removes a toolbar from the display.

id: Must be one of "top," "bottom," "left", "right," or "all."

11.11.3 OIT.toolbars.show

```
OIT.toolbars.show( id )
```

Shows an existing toolbar.

id: Must be one of "top," "bottom," "left", "right," or "all."

11.11.4 OIT.toolbars.hide

```
OIT.toolbars.hide( id )
```

Hides a toolbar.

id: Must be one of "top," "bottom," "left", "right," or "all."

11.11.5 OIT.toolbars.setsize

```
OIT.toolbars.hide( id, size )
```

Sets the size of a toolbar.

id: Must be one of "top," "bottom," "left", or "right."

size: Either a number of pixels for the new size value, or "auto." If the size is set to "auto," the element containing the toolbar content (the element parameter provided to OIT.toolbars.add) must have a size specified via CSS.

11.11.6 OIT.toolbars.addEventListener

```
OIT.toolbars.addEventListener( name, fn )
```

Adds an event listener for OIT toolbar events. The event listener function receives the event as its argument. event.preventDefault() may be called to prevent the default behavior described below.

- *name*: The name of the event to be monitored.
- *fn*: A Javascript function that will be called with event information when the specified event occurs.

See Events from the Javascript API for information on how events are handled.

The name value may be any of the following:

Event Name	Description	Event Detail	Default Behavior
add	Called after a toolbar has been successfully added to an edge of the web view.	**id**: The toolbar id **edge**: The edge on which the new toolbar is placed ("top", "bottom", "left", or "right").	(none)
remove	Called after a toolbar has been removed from an edge of the web view.	**id**: The toolbar id **edge**: The edge on which the toolbar had been placed ("top", "bottom", "left", or "right").	(none)
resize	Called after a toolbar has been resized.	**id**: The toolbar id **edge**: The edge on which the new toolbar is located ("top", "bottom", "left", or "right"). **width**: The new width of the toolbar, in pixels. **height**: The new height of the toolbar, in pixels.	(none)

Event Name	Description	Event Detail	Default Behavior
show	Called after a toolbar has been shown.	**id**: The toolbar id **edge**: The edge on which the toolbar is located ("top", "bottom", "left", or "right"). **width**: The current width of the toolbar, in pixels. **height**: The current height of the toolbar, in pixels.	(none)
hide	Called after a toolbar has been hidden	**id**: The toolbar id **edge**: The edge on which the toolbar is located ("top", "bottom", "left", or "right").	(none)

Example

```
OIT.toolbars.addEventListener( "resize", function(event)
  {
    if( event.detail.edge == "top" )
       alert( "The top toolbar is now " + event.detail.width + " pixels wide." );
  });
```

11.11.7 OIT.toolbars.removeEventListener

```
OIT.toolbars.removeEventListener( name, fn )
```

Removes an event listener for OIT toolbar events.

name: The name of the event that was being monitored.

fn: The Javascript function to be removed from notifications of the specified event.

See Events from the Javascript API for information on how events are handled.

11.12 OIT.local

Changes to localization strings must be made before the window "load" event fires, so changes must be made by Javascript code that executes when the document loads. Changes made to OIT.local after that point may not have any effect.

For instance, to use German language strings in the Web View Export UI, the customer would have top level Javascript code like this:

OIT.local.AUTHOR = "Verfasser"; (etc.)

Field	Default Value
ADD	"add"
ADD_COMMENT	"Add comment"
ADD_COMMENTS	"Add comments"
ADD_HIGHLIGHTS	"Add highlights"
AUTHOR	"Author"
BYTES	"Bytes"
CANCEL	"Cancel"
CATEGORY	"Category"

Field	Default Value
CHARACTERS	"Characters"
CHARACTERS_WITH_SPACES	"Characters (with spaces)"
CLOSE	"Close"
COMPANY	"Company"
COMPLETED_DATE	"Completed Date"
CREATED	"Created"
DOCUMENT_PROPERTIES	"Document properties"
EDIT_COMMENT	"Edit comment"
EMAIL_ATTACHMENTS	"Attachments:"
EMAIL_CC	"CC:"
EMAIL_FROM	"From:"
EMAIL_TO	"To:"
EMAIL_TOPIC	"Topic:"
EMAIL_PRIORITY	"Priority:"
EMAIL_IMPORTANCE	"Importance:"
EMAIL_OWNER	"Owner:"
EMAIL_STATUS	"Status:"
EMAIL_PERCENT	"Percent Complete:"
EMAIL_STARTDATE	"Start Date:"
EMAIL_ENDDATE	"End Date:"
EMAIL_CREATED	"Created:"
EMAIL_CATEGORY	"Categories:"
EMAIL_FIRST	"First:"
EMAIL_LAST	"Last:"
EMAIL_MIDDLE	"Middle:"
EMAIL_SUFFIX	"Suffix:"
EMAIL_TITLE	"Job Title:"
EMAIL_COMPANY	"Company:"
EMAIL_JOBTITLE	"Job Title:"
EMAIL_BPHONE	"Business Phone:"
EMAIL_BADDRESS	"Business Address:"
EMAIL_FAX	"Business Fax:"
EMAIL_EMAIL	"Email Address:"
EMAIL_HPHONE	"Home Phone:"
EMAIL_HADDRESS	"Home Address:"
EMAIL_MPHONE	"Mobile:"
EMAIL_IM	"IM:"
EMAIL_WEBPAGE	"Webpage:"

Field	Default Value
EMAIL_ENTRYTYPE	"Entry Type:"
EMAIL_STARTTIME	"Start Time:"
EMAIL_ENDTIME	"End Time:"
EMAIL_DURATION	"Duration:"
EMAIL_REQATTEND	"Required Attendees:"
EMAIL_OPTATTEND	"Optional Attendees:"
EMAIL_LOCATION	"Location:"
EMAIL_RECURTYPE	"Recurrence Type:"
EMAIL_RECURRANGE	"Recurrence Range:"
FIT_PAGE	"Fit page"
FIT_PAGE_TO_WINDOW	"Fit page to window"
FIT_TO_WINDOW_WIDTH	"Fit page to window width"
FIT_WIDTH	"Fit width"
GREEN_HIGHLIGHT	"Green highlight"
HIDDEN_SLIDES	"Hidden Slides"
HIGHLIGHT	"Highlight"
KEYWORDS	"Keywords"
LAST_SAVED_BY	"Last saved by"
LINES	"Lines"
LOADING	"Loading"
MESSAGES	"Messages"
MODIFIED	"Modified"
NEXT_HIGHLIGHT	"Next highlight"
NOTES	"Notes"
OK	"OK"
OTHER_PROPERTIES	"Other Properties"
PAGE	"Page "
PAGE_DOWN	"Next Page"
PAGE_UP	"Previous Page"
PAGES	"Pages"
PARAGRAPHS	"Paragraphs"
PINK_HIGHLIGHT	"Pink highlight"
PLEASE_WAIT	"Please wait"
PREPARING_TO_PRINT	"Preparing to print..."
PREVIOUS_HIGHLIGHT	"Previous highlight"
PRINT_DATE	"Print Date"
RECT_SELECTION_MODE	"Rectangular selection mode"
REMOVE	"remove"

Field	Default Value
REMOVE_CURRENT_HIGHLIGHT	"Remove current highlight"
REVISION_DATE	"Revision date"
REVISION_NUMBER	"Revision number"
ROTATE_LEFT	"Rotate left"
ROTATE_RIGHT	"Rotate right"
SEARCHING	"Searching..."
SHOWATTACHMENTS	"Show Attachments"
SHOWDETAILS	"Show Details"
STATISTICS	"Statistics"
STATUS	"Status"
SUBJECT	"Subject"
TEXT_SELECTION_MODE	"Text selection mode"
TITLE	"Title"
TOTAL_EDITING_TIME	"Total editing time"
VERSION_DATE	"Version date"
VERSION_NUMBER	"Version number"
WORDS	"Words"
YELLOW_HIGHLIGHT	"Yellow highlight"
ZOOM	"Zoom "
ZOOM_IN	"Zoom in"
ZOOM_OUT	"Zoom out"
ZOOM_TO_ACTUAL_SIZE	"Zoom to actual size"

11.13 Events from the Javascript API

The Web View Export API allows scripts to register Javascript event handlers for certain events that occur while a document is being displayed. Instead of directly exposing the HTML DOM for registering event handlers, the API provides its own addEventListener methods for certain categories of objects, defined by Web View Export's own internal document model. The prototype is the same for all of these objects:

```
OIT.category.addEventListener( name, fn )
```

For every one of these functions, there is a corresponding:

```
OIT.category.removeEventListener( name, fn )
```

All Events are Custom Events

All events emitted by the Web View Export API are Javascript custom events, even when caused by a standard user action such as a mouse click event. The particular data associated with a Web View Export event is available from the "detail" object that is associated with the event object.

Example: Processing Events

In this example we react to the "document load" event by showing an alert box to indicate the URL of the newly loaded document. The url is contained in evt.detail.

```
 OIT.document.addEventListener( "load", function(evt) {  alert( "Just loaded " +
evt.detail.url );
  });
```

Example: Preventing Default Behavior

If there is default behavior in the web view for a particular event, you can prevent it from occurring by calling the preventDefault() method on the event object.

```
 OIT.hyperlinks.addEventListener( "click", function(evt) {
   if( evt.detail.external ) /* don't follow external links */
     evt.preventDefault();
 });
```

Example: Events from DOM Events

Though all of the events defined by the Web View Export API are custom events, some events may be triggered in response to DOM events whose original event information is worth providing to your script. In these cases, the detail object will include the information from the original event object, as srcEvent.

```
OIT.hyperlinks.addEventListener( "click", function(evt) {
  alert( "Click event at x,y =" + evt.detail.srcEvent.clientX + ", " +
evt.detail.srcEvent.clientY  );
  });
```

Part IV

Using the Java API

This section provides details about using the Web View Export SDK with the Java API.

Part IV contains the following chapters:

12

Web View Export Java API

The Java API is an add-on to the Outside In Export SDKs that enables developers to use Java to create applications using Outside In Technology.

12.1 Requirements

To use the API, the following set of modules and tools are required:

- Java JDK 6 or later
- The Outside In Web View developer's redistributable modules
- The API libraries:
  - oilink.jar - The Java library to access the Outside In technologies
  - oilink (on Unix)/oilink.exe (on Windows) - The bridge modules between Java and the C-APIs.

All of the Outside In modules should be in the same directory as oilink.jar.

The SDK includes sample source code to demonstrate how such web applications may be written. These sample applications are written as simply and generically as possible, and will not fill all of the needs of your particular application. They are intended for instructional purposes only.

12.2 Getting Started

There are two steps in developing applications using the APIs. In the first step, you configure the environment to create your application (typical programming tasks not directly related to these APIs); and in the second step, you generate code to utilize the functionality of these libraries.

12.2.1 Configure the Environment

To set up the environment to create a Java application, you need to add the oilink.jar library to your project. (This can be done in Eclipse in the Project Properties dialog by selecting *Java Build Path properties > Libraries tab > Add external JARs > browse to oilink.jar*.)

12.2.2 Generate Code

Sample application code included with the SDK, WebViewSample, is a minimal demonstration of how to use this API.

All the functionality required to perform a conversion is provided in an Exporter object. The basic process of exporting a file involves the following tasks:

1. Create an Exporter object.
2. Configure the export.
3. Set the source and primary destination files.
4. Set the output type.
5. (Optional) Provide a callback handler.
6. Run the export.

Tasks 2 through 5 can be done in any order between the first and last task.

12.2.2.1 Create an Exporter Object

To obtain access to the Outside In functionality, you should call the utility function in the "OutsideIn" class. This will provide you an instance of an Exporter Object.

```
Exporter exporter = OutsideIn.newLocalExporter();
```

12.2.2.2 Configure the Output

The Outside In API is highly configurable, and presents numerous options to fine-tune the way a document is exported. Each option has a "set" and "get" method to set or retrieve the currently set value.

```
exporter.setPerformExtendedFI(true);
int timezoneOffset = exporter.getTimeZoneOffset();
```

12.2.2.3 Set the Source and Primary Destination Files

You are required to specify the source file and the destination file. This is done similarly to setting options using "set" methods.

```
exporter.setSourceFile(inputFile);
exporter.setDestinationFile(outputFile);
```

There are other options that can be set at this time to specify the way to handle the input file, such as providing a SourceFormat to provide a mechanism to handle the input file in a different format than that which it is identified as.

The API also supports opening certain types of embedded documents from within an input file. For example, a .zip file may contain a number of embedded documents; and an email message saved as a .msg file may contain attachments. The API provides the means of opening these types of embedded documents. This can be done by opening the parent document and then the embedded document can be opened through this exporter object.

```
// subdocId is the sequential number of the node in the archive file
Exporter exporterNode = exporter.newTreeNodeExporter(subdocId);
```

12.2.2.4 Set the Output Type

In this step, you specify the output format.

```
exporter.setDestinationFormat(FileFormat.FI_HTML5);
```

12.2.2.5 Provide a Callback Handler

Outside In Technology provides callbacks that allow the developer to intervene at critical points in the export process. To respond to these callbacks, you have to

subscribe to any messages that you are interested in by overriding the message handlers from the Callback class. After creating an object of this class, set the callback option to this object and the messages will be passed to your object.

```
class CallbackHandler extends Callback
{
  … // implementation of messages to handle - described in the API documentation
}
CallbackHandler callback = new CallbackHandler();
exporter.setCallbackHandler(callback);
```

12.2.2.6 Run the Export

After all the previous steps are completed, you can produce the desired output.

```
exporter.export();
```

13

Web View Export Java Classes

This chapter describes the Web View Export Java classes.

13.1 Annotation Class

Annotation is an abstract base class for the Annotation objects.

Accessors

- Height (long) Height of area in coordinates or rows

```
void setHeight(long)
long getHeight()
```

- Left (long) Leftmost coordinate or column

```
void setLeft(long)
long getLeft()
```

- Opacity (float) Opacity of the annotation. Range 0.0 - 1.0; setting opacity to 0 makes the annotation invisible

```
void setOpacity(float) throws Exception
float getOpacity()
```

- SectionIndex (long) 0-based page/sheet/image/slide index

```
void setSectionIndex(long)
long getSectionIndex()
```

- Top (long) Top coordinate or row

```
void setTop(long)
long getTop()
```

- Units (Annotation.UnitTypeValue) Type of units in which Height, Width, Left and Top are described in

```
void setUnits(Annotation.UnitTypeValue)
Annotation.UnitTypeValue getUnits()
```

- UserId (long) User Data

```
void setUserId(long)
long getUserId()
```

- Width (long) Width of area in coordinates or columns

```
void setWidth(long)
```

```
long getWidth()
```

Annotation.UnitTypeValue Enumeration

The UnitTypeValue is an enumeration of the various unit types that annotation positions can be described in.

- PIXELS: Units specified in Pixels
- TWIPS: Units specified in Twips (1/1440th of an inch)
- CELLS: Units specified in cell positions

13.2 Callback Class

Callback messages are notifications that come from Outside In during the export process, providing information and sometimes the opportunity to customize the generated output. To access callback messages, your code must create an object that inherits from Callback and pass it through the API's setCallbackHandler method. Your object can implement methods that override the default behavior for whichever methods your application is interested in.

Callback has two methods that you can override: createNewFile and newFileInfo.

13.2.1 createNewFile

```
CreateNewFileResponse createNewFile( FileFormat parentOutputId,  FileFormat
outputId,
   AssociationValue association, String path)  throws IOException
```

This callback is made any time a new output file needs to be generated. This gives the developer the chance to affect where the new output file is created, how it is named, and the URL (if any) used to reference the file.

Parameters

- parentOutputId: File format identifier of the parent file
- outputId: File format identifier of the file created
- association: An AssociationValue that describes relationship between the primary output file and the new file.
- path: Full path of the file to be created

Return Value

To take action in response to this notification, return a CreateNewFileResponse object with the new file information. If you wish to accept the defaults for the path and URL, you may return null.

CreateNewFileResponse Class

This is a class to define a new output file location in response to a CreateNewFile callback. If you do not wish to change the path to the new output file, you may use the path as received. If you do not wish to specify the URL for the new file, you many specify it as null.

Constructor

```
CreateNewFileResponse(File file, String url)
```

- file: File object containing the full path to new file
- url: A new URL that references the newly created file. This parameter can be null.

AssociationValue Enumeration

This enumeration defines, for a new file created by an export process, the different possible associations between the new file and the primary output file. Its value may be one of the following:

- ROOT - indicates the primary output file
- CHILD - a new file linked (directly or indirectly) from the primary output file
- SIBLING - indicates new files not linked from the primary output file
- COPY - the file was copied as a part of a template macro operation.
- REQUIREDNAME - not used

Note that some of these relationships will not be possible in all Outside In Export SDKs.

13.2.2 newFileInfo

```
void newFileInfo( FileFormat parentOutputId, FileFormat outputId,
    AssociationValue association, String path, String url) throws IOException
```

This informational callback is made just after each new file has been created.

Parameters

- parentOutputId: File format identifier of the parent file
- outputId: File format identifier of the file created
- association: An AssociationValue that describes relationship between the primary output file and the new file.
- path: Full path of the file created
- url: URL that references the newly created file

Example

Here is a basic callback handler that notifies an application that it has received newFileInfo notifications.

```
 public static class CallbackHandler extends Callback
 {
   myApplication m_theApp;

   public CallbackHandler( myApplication app )
   {
     m_theApp = app;
   }

@Override public void newFileInfo(FileFormat parentOutputId,
       FileFormat outputId, AssociationValue association,
       String path, String url) throws IOException
   {
     if( association == AssociationValue.ROOT )
       m_theApp.primaryOutputIsReady(true);

     m_theApp.newOutputFile(path);
```

```
    }
}
```

13.3 ColorInfo Class

ColorInfo is a class to define a color or to use a default color in appropriate cases.

Constructors

```
ColorInfo()
```

Initializes an ColorInfo object to use the default color.

```
public ColorInfo(Byte red,
   Byte green,
   Byte blue)
```

Initializes an ColorInfo object with the specified RGB values.

Properties

- Blue (Byte) - Blue component of the color
- Green (Byte) - Green component of the color
- Red (Byte) - Red component of the color
- UseDefault (Byte) - Set to true if the default color is used

13.4 Exporter Interface

This section describes the properties and methods of Exporter.

All of Outside In's Exporter functionality can be accessed through the Exporter Interface. The object returned by OutsideIn class is an implementation of this interface. This class derives from the Document Interface, which in turn is derived from the OptionsCache Interface.

Methods

- **addKeyValuePairs**

  ```
  void addKeyValuePairs(
  Map<String, String> pairs)
  ```

  This function lets you specify key/value pairs for use in the exported document.

  pairs: Key value pairs where the key is the name of the data.

- **addStampAnnotation**

  ```
  void addStampAnnotation(
       StampAnnotation StampAnno
  )
  ```

  Add a stamp annotation to the exported document.

  StampAnno: A StampAnnotation object with information about the stamp to add.

- **addStampAnnotation Properties**

  ```
  void addStampAnnotation(
        StampAnnotation StampAnno,
        Map<String, String> Properties)
  ```

Add a stamp annotation to the exported document and add properties associated with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add.

Properties: Key value pairs of name/value of properties associated with this annotation.

- **addStampAnnotation Comment**

```
void addStampAnnotation(
      StampAnnotation StampAnno,
      String Comment)
```

Add a stamp annotation to the exported document and associate a comment with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add

Comment: Comment text to associate with the annotation

- **addStampAnnotation Comment and Properties**

```
void addStampAnnotation(
      StampAnnotation StampAnno,
      String Comment,
      Map<String, String> Properties)
```

Add a stamp annotation to the exported document and associate a comment and properties with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **addStampImageFile**

```
void addStampImageFile(
      Map<String, File> stamps
)throws OutsideInException
```

This method maps a stamp name to an image file.

stamps: Key value pairs of stamp names and image files

- **addStampImageURL**

```
void addStampImageURL(
      Map<String, URI> stamps
)
```

This method maps a stamp name to an image URL.

stamps: Key value pairs of stamp names and image URLs

- **applyHighlights**

```
void applyHighlights(
      String HighlightJsonData
)
```

This method applies a set of highlights from a JSON-encoded text stream previously generated from the Web View Javascript library.

HighlightJsonData: stream of highlight (and comment) information that was obtained via the Web View Javascript API function OIT.highlights.serialize.

- **getExportStatus**

```
ExportStatus getExportStatus()
```

This function is used to determine if there were conversion problems during an export. The ExportStatus object returned may have information about sub-document failures, areas of a conversion that may not have high fidelity with the original document. When applicable the number of pages in the output is also provided.

- **addAreaHighlight**

```
void addAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno
)
```

Highlight an area on a page in a document.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

- **addAreaHighlight and Add Annotation Properties**

```
void addAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      Map<String, String> Properties
)
```

Add an area highlight to a document and associate properties with the annotation.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Properties: Key value pairs of name/value of properties associated with this annotation

- **addAreaHighlight and Associate Comment with Annotation**

```
void addAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      String Comment
)
```

Add an area highlight to a document, and associate a comment.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Comment: Comment text to associate with the annotation

- **addAreaHighlight with Comment and Properties**

```
void addAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      String Comment,
      Map<String, String> Properties
)
```

Add an area highlight to a document, including associated comment text and properties.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **addTextHighlight**

```
void addTextHighlight(
      HighlightTextAnnotation textanno
)
```

Highlight text in a document.

textanno: A HighlightTextAnnotation object with information about the text to highlight

- **addTextHighlight and Add Annotation Properties**

```
void addTextHighlight(
      HighlightTextAnnotation textanno,
      Map<String, String> Properties
)
```

Highlight text in a document and associate properties with the annotation.

textanno: A HighlightTextAnnotation object with information about the text to highlight

Properties: Key value pairs of name/value of properties associated with this annotation

- **addTextHighlight and Associate a Comment**

```
void addTextHighlight(
      HighlightTextAnnotation textanno,
      String Comment
)
```

Highlight text in a document and associate a comment with the highlight.

textanno: A HighlightTextAnnotation object with information about the text to highlight

Comment: Comment text to associate with the annotation

- **addTextHighlight with Comment and Properties to Annotation**

```
void addTextHighlight(
      HighlightTextAnnotation textanno,
      String Comment,
      Map<String, String> Properties
)
```

Highlight text in a document and provide comment text and properties to be associated with the annotation.

textanno: A HighlightTextAnnotation object with information about the text to highlight

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **newSubdocumentExporter**

```
Exporter newSubdocumentExporter(
      int SubdocId,
      SubDocumentIdentifierTypeValue idType
) throws Exception
```

Create a new Exporter for a subdocument.

SubDocId: Identifier of the subdocument

idType: Type of subdocument

SubDocumentIdentifierTypeValue: This is an enumeration for the type of subdocument being opened.

- XMLEXPORTLOCATOR: Subdocument to be opened is based on output of XML Export (SubdocId is the value of the object_id attribute of a locator element.)
- ATTACHMENTLOCATOR: Subdocument to be opened is based on the locator value provided by the one of the Export SDKs.
- EMAILATTACHMENTINDEX: Subdocument to be opened is based on the index of the attachment from an email message. (SubdocId is the zero-based index of the attachment from an email message file. The first attachment presented by OutsideIn has the index value 0, the second has the index value 1, etc.)

Returns: A new Exporter object for the subdocument

- **newTreeNodeExporter**

```
Exporter newTreeNodeExporter(
      int dwRecordNum
) throws OutsideInException
```

Create a new Exporter for an archive node.

dwRecordNum: The number of the record to retrieve information about. The first node is node 0.

Returns: A new Exporter object for the archive node

- **newTreeNodeExporter with Search Export Data**

```
Exporter newTreeNodeExporter(
      int flags,
      int params1,
      int params2
) throws OutsideInException
```

Create a new Exporter for an archive node. To use this function, you must first process the archive with Search Export and save the Node data for later use in this function.

Flags: Special flags value from Search Export

Params1: Data1 from Search Export

Params2: Data2 from Search Export

Returns: A new Exporter object for the archive node

- **export**

```
void export() throws OutsideInException
```

Perform the conversion.

- **setDestinationFile**

```
OptionsCache setDestinationFile(
      File file
) throws OutsideInException
```

Set the location of the destination file

file: The destination file to which the export will be written

Returns: The updated options object

13.4.1 Document Interface

All of the Outside In document-related methods are accessed through the Document Interface.

Methods

- **close**

```
void close() throws IOException
```

Closes the currently open document.

- **getArchiveNodeCount**

```
int getArchiveNodeCount() throws OutsideInException
```

Retrieves the number of nodes in an archive file.

Returns the number of nodes in the archive file or 0 if the file is not an archive file.

- **getFileId**

```
FileFormat getFileId(FileIdInfoFlagValue dwFlags) throws OutsideInException
```

Gets the format of the file based on the technology's content-based file identification process.

dwFlags: Option to retrieve the file identification pre-Extended or post-Extended Test

Returns the format identifier of the file.

- **getObjectInfo**

```
ObjectInfo getObjectInfo() throws OutsideInException
```

Retrieves the information about an embedded object.

Return: An ObjectInfo object with the information about the embedded object

- **getRecordInfo**

```
TreeRecord getRecordInfo(int nNodeNum) throws OutsideInException
```

Retrieves information about a record in an archive file

nNodeNum: The number of the record to retrieve information about. The first node is node 0.

Return Value: A TreeRecord object with the information about the record

- **saveRecord**

```
void saveRecord(
      int nNodeNum,
      File file) throws OutsideInException
```

  Extracts a record in an archive file to disk.

  nNodeNumType: The number of the record to retrieve information about. The first node is node 0.

  file: The destination file to which the file will be extracted.

- **SaveRecord with Search Export Flags**

```
void saveRecord(
      int flags,
      int params1,
      int params2,
      File file) throws OutsideInException
```

  Extracts a record in an archive file to disk without reading the data for all nodes in the archive in a sequential order. To use this function, you must first process the archive with Search Export and save the Node data for later use in this function.

  flagsType: Special flags value from Search Export

  params1: Data1 from Search Export+

  params2: Data2 from Search Export

  file: The destination file to which the file will be extracted

- **setSourceFile**

```
OptionsCache setSourceFile( File file) throws OutsideInException
```

  Set the source document.

  file: The source file to export

  Returns: The options cache object associated with this document

13.4.2 OptionsCache Class

This section describes the OptionsCache class.

The options that configure the way outputs are generated are accessed through the OptionsCache class.

All of the options described in the following subsections are available through this interface. Other methods in this interface are described below.

Methods

- `OptionsCache setSourceFile(File file) throws OutsideInException`

  Sets the source document to be opened.

  file: Full path to source file

- `OptionsCache setSourceFormat(FileFormat fileId)`

  Sets the source format to process the input file as, ignoring the algorithmic detection of the file type.

fileId: the format to treat the input document as.

- `OptionsCache setDestinationFile(File file) throws OutsideInException`

  Sets the location of the destination file.

  file: Full path to the destination file

- `OptionsCache setDestinationFormat(FileFormat fileId)`

  Sets the destination file format to which the file should be converted.

  fileId: the format to convert the input document(s) to.

- `OptionsCache setCallbackHandler(Callback callback)`

  Sets the object to use to handle callbacks.

  callback: the callback handling object.

- `OptionsCache setPasswordsList(List<String> Passwords)`

  Provides a list of strings to use as passwords for encrypted documents. The technology will cycle through this list until a successful password is found or the list is exhausted.

  Passwords: List of strings to be used as passwords.

- `OptionsCache setLotusNotesId(String NotesIdFile)`

  Sets the Lotus Notes ID file location.

  NotesIdFile: Full path to the Notes ID file.

13.4.2.1 BiDiReorderMethod

OIT Option ID: SCCOPT_REORDERMETHOD

This option controls how the technology reorders bidirectional text.

Data Type

BiDiReorderMethodValue

BiDiReorderMethodValue Enumeration

- UNICODEOFF: This disables any processing for bidirectional characters. This option is the default.
- UNICODELTOR: Characters displayed using the Unicode bidirectional algorithm assuming a base left-to-right order. Use this option to enable bidirectional UNICODERTOL.
- RIGHTTOLEFT: Characters displayed using the Unicode bidirectional algorithm assuming a base right-to-left order. Use this option to force starting bidirectional rendering in the right-to-left order.

Default

BiDiReorderMethodValue.UNICODEOFF

13.4.2.2 DefaultInputCharacterSet

OIT Option ID: SCCOPT_DEFAULTINPUTCHARSET

This option is used in cases where Outside In cannot determine the character set used to encode the text of an input file. When all other means of determining the file's

character set are exhausted, Outside In will assume that an input document is encoded in the character set specified by this option. This is most often used when reading plain-text files, but may also be used when reading HTML or PDF files.

Data Type

DefaultInputCharacterSetValue

DefaultInputCharacterSetValue Enumeration

DefaultInputCharacterSetValue can be one of the following enumerations:

SYSTEMDEFAULT

UNICODE

BIGENDIANUNICODE

LITTLEEENDIANUNICODE

UTF8

UTF7

ASCII

UNIXJAPANESE

UNIXJAPANESEEUC

UNIXCHINESETRAD1

UNIXCHINESEEUCTRAD1

UNIXCHINESETRAD2

UNIXCHINESEEUCTRAD2

UNIXKOREAN

UNIXCHINESESIMPLE

EBCDIC37

EBCDIC273

EBCDIC274

EBCDIC277

EBCDIC278

EBCDIC280

EBCDIC282

EBCDIC284

EBCDIC285

EBCDIC297

EBCDIC500

EBCDIC1026

DOS437

DOS737

DOS850

DOS852

DOS855

DOS857

DOS860

DOS861

DOS863

DOS865

DOS866

DOS869

WINDOWS874

WINDOWS932

WINDOWS936

WINDOWS949

WINDOWS950

WINDOWS1250

WINDOWS1251

WINDOWS1252

WINDOWS1253

WINDOWS1254

WINDOWS1255

WINDOWS1256

WINDOWS1257

ISO8859_1

ISO8859_2

ISO8859_3

ISO8859_4

ISO8859_5

ISO8859_6

ISO8859_7

ISO8859_8

ISO8859_9

MACROMAN

MACCROATIAN

MACROMANIAN

MACTURKISH

MACICELANDIC

MACCYRILLIC

MACGREEK

MACCE

MACHEBREW

MACARABIC

MACJAPANESE

HPROMAN8

BIDIOLDCODE

BIDIPC8

BIDIE0

RUSSIANKOI8

JAPANESEX0201

Default

DefaultInputCharacterSetValue.SYSTEMDEFAULT

13.4.2.3 DefaultPageMargins

This option specifies the top, left, bottom and right margins in twips from the edges of the page. For instance, setting all the values to 1440 creates a 1-inch margin on all sides. Page margins will only be applied when formatting word processing, database and spreadsheet files.

Please note all margins are applied before scaling with the PageFitMode option. This option is overridden when the UseDocumentPageSettings option is set to true and print margins are specified in the input document. This option does not affect the output of bitmap, presentation, vector or archive files.

Data Type

Margins

Data

A Margins object with the margins on the 4 sides defined.

Default

1 inch for all margins

13.4.2.4 DefaultPageSize

This option allows the developer to specify the size of each page in the generated output file. The size may be specified in inches, points, centimeters or picas. This option is only valid when UseDocumentPageSettings is set to false. 1 inch = 6 picas = 72 points = ~ 2.54 cm.

Data Type

PageInfo

Data

A PageInfo object with the page size information.

Default
8.5 inches by 11 inches

13.4.2.5 DefaultRenderFont
OIT Option ID: SCCOPT_DEFAULTPRINTFONT

This option sets the font to use when the font specified by the filter is not available on the system.

Data Type
FontInfo

Data
The default font to be used

Default
None

13.4.2.6 DocumentMemoryMode
OIT Option ID: SCCOPT_DOCUMENTMEMORYMODE

This option determines the maximum amount of memory that the internal storage may use to store the document's data, from 4 MB to 1 GB. The more memory the internal storage has available to it, the less often it needs to re-read data from the document.

Data
- SMALLEST: 1 - 4MB
- SMALL: 2 - 16MB
- MEDIUM: 3 - 64MB
- LARGE: 4 - 256MB
- LARGEST: 5 - 1 GB

Default
DocumentMemoryModeValue.SMALL

13.4.2.7 EmailAttachmentHandling
This option determines whether email attachments are converted or extracted.

Data Type
EmailAttachmentHandlingValue

EmailAttachmentHandlingValue Enumeration
EmailAttachmentHandlingValue can be one of the following enumerations:

- NONE: Attachments will not be converted or extracted.
- EXPORT: The attachments to this email will be converted.
- EXPORTRECURSIVE: The attachments to this email will be converted; if any are emails, they will also have their attachments converted, and so on.

- EXTRACT: Attachments will be extracted in their native form, stored alongside the Web View output files.

Default

EmailAttachmentHandlingValue.NONE

13.4.2.8 ExternalStylesheets

String option for specifying the url of a user supplied stylesheet to be included in the output html file. Values specified through this option will appear in output HTML files as linked stylesheets.

Example

```
<script src="oit.css">
<link rel="stylesheet" href="/some/path/myStylesheet.css"/>
```

Data Type

List<String>

Default

None

13.4.2.9 FallbackFormat

This option controls how files are handled when their specific application type cannot be determined. This normally affects all plain-text files, because plain-text files are generally identified by process of elimination, for example, when a file isn't identified as having been created by a known application, it is treated as a plain-text file. It is recommended that None be set to prevent the conversion from exporting unidentified binary files as though they were text, which could generate many pages of "garbage" output.

Data Type

FallbackFormatValue

FallbackFormatValue Enumeration

FallbackFormatValue can be one of the following enumerations:

- TEXT: Unidentified file types will be treated as text files.
- NONE: Outside In will not attempt to process files whose type cannot be identified.

Default

FallbackFormatValue.TEXT

13.4.2.10 FontAliasList

This option enables the capability to specify which font on the system should be used when a specific font referenced in the original file is not available. A different alias can be set for each font desired to be mapped.

Data Type

FontAliases

Data
A FontAliases object with a list of font matchings

Default
Please refer to the section "SCCOPT_PRINTFONTALIAS" in the C-API for the default font alias list.

13.4.2.11 FontBaseURL
This option allows the developer to specify the base URL used in the Web View Export output when referencing a non-exported font as a downloadable font. (Exported fonts will be placed in the exported output location.)

Data Type
String

Default
None

13.4.2.12 FontDirectories
This option allows the developer to specify one or more font directories where fonts are located for use by Web View Export. If multiple font directories are specified, they should be delimited by a colon on Linux and UNIX systems and a semi-colon on Windows systems. Please note that Web View Export supports single TrueType fonts (*.ttf, *.TTF) and TrueType collections (*.ttc, *.TTC), not Windows bitmap fonts (*.fon, *.FON), or any other type of font. Also, Web View Export does not require case-sensitive font filenames on UNIX systems.

Data Type
List<File>

Default
None

13.4.2.13 FontFilter
This option allows the developer to specify a list of fonts to be included or excluded during the export process.

Data Type
FontList

Data
A FontFilter object describing the inclusion or exclusion list.

Default
None

13.4.2.14 FontReferenceMethod
This option controls the way embedded fonts are presented (or not) in the HTML5 output.

Data Type

FontReferenceMethodValue

FontReferenceMethodValue Enumeration

- REFERENCEBYNAME: The font is referenced in the output file by name only
- REFERENCEEXPORTED: The font is subsetted, exported during document conversion, and referenced in the output by file name
- REFERENCEBYBASEURL: The full font is referenced in the output using the font file name and the base URL supplied by the FontBaseURL option.

Default

FontReferenceMethodValue.REFERENCEBYNAME

13.4.2.15 GraphicHeightLimit

OIT Option ID: SCCOPT_GRAPHIC_HEIGHTLIMIT

Note that this option differs from the behavior of setting the height of graphics in that it sets an upper limit on the image height. Images larger than this limit will be reduced to the limit value. However, images smaller than this height will not be enlarged when using this option. Setting the height using GraphicHeight causes all output images to be reduced or enlarged to be of the specified height.

A value of zero is equivalent to no limit, which causes this option to be ignored.

Data Type

long

Default

0

13.4.2.16 GraphicOutputDPI

OIT Option ID: SCCOPT_GRAPHIC_OUTPUTDPI

This option allows the user to specify the output graphics device's resolution in DPI and only applies to images whose size is specified in physical units (in/cm). For example, consider a 1" square, 100 DPI graphic that is to be rendered on a 50 DPI device (GraphicOutputDPI is set to 50). In this case, the size of the resulting TIFF, BMP, JPEG, GIF, or PNG will be 50 x 50 pixels.

In addition, setting this option to 0(Maintain Source Image DPI), can be used to suppress any dimensional changes to an image. In other words, a 1" square, 100 DPI graphic will be converted to an image that is 100 x 100 pixels in size. This value indicates that the DPI of the output device is not important. It extracts the maximum resolution from the input image with the smallest exported image size.

Setting this option to 0 may result in the creation of extremely large images. Be aware that there may be limitations in the system running this technology that could result in undesirably large bandwidth consumption or an error message. Additionally, an out of memory error message will be generated if system memory is insufficient to handle a particularly large image.

Also note that setting this option to 0 will force the technology to use the DPI settings already present in raster images, but will use the current screen resolution as the DPI setting for any other type of input file.

For some output graphic types, there may be a discrepancy between the value set by this option and the DPI value reported by some graphics applications. The discrepancy occurs when the output format uses metric units (DPM, or dots per meter) instead of English units (DPI, or dots per inch). Depending on how the graphics application performs rounding on meters to inches conversions, the DPI value reported may be 1 unit more than expected. An example of a format which may exhibit this problem is PNG. The maximum value that can be set is 2400 DPI.

Data Type

long

Default

96

13.4.2.17 GraphicSizeLimit

OIT Option ID: SCCOPT_GRAPHIC_SIZELIMIT

This option is used to set the maximum size of the exported graphic in pixels. It may be used to prevent inordinately large graphics from being converted to equally cumbersome output files, thus preventing bandwidth waste.

This setting takes precedence over all other options and settings that affect the size of a converted graphic.

When creating a multi-page TIFF file, this limit is applied on a per page basis. It is not a pixel limit on the entire output file.

Data Type

long

Default

0

13.4.2.18 GraphicWidthLimit

OIT Option ID: SCCOPT_GRAPHIC_WIDTHLIMIT

This option allows a hard limit to be set for how wide in pixels an exported graphic may be. Any images wider than this limit will be resized to match the limit. It should be noted that regardless whether the GraphicHeightLimit option is set or not, any resized images will preserve their original aspect ratio.

Note that this option differs from the behavior of setting the width of graphics by using GraphicWidth in that it sets an upper limit on the image width. Images larger than this limit will be reduced to the limit value. However, images smaller than this width will not be enlarged when using this option. Setting the width using GraphicWidth causes all output images to be reduced or enlarged to be of the specified width.

Data Type

long

Default

0

13.4.2.19 IECondCommentMode

OIT Option ID: SCCOPT_HTML_COND_COMMENT_MODE

Some HTML input files may include "conditional comments", which are HTML comments that mark areas of HTML to be interpreted in specific versions of Internet Explorer, while being ignored by other browsers. This option allows you to control how the content contained within conditional comments will be interpreted by Outside In's HTML parsing code.

Data Type

IECondCommentFlagValues

IECondCommentFlagValues Enumeration

IECondCommentFlagValues can be one or more of the following enumerations ORed together:

- NONE: Don't output any conditional comment
- IE5: Include the IE5 comments
- IE6: Include the IE6 comments
- IE7: Include the IE7 comments
- IE8: Include the IE8 comments
- IE9: Include the IE9 comments
- ALL: Include all conditional comments

Default

IECondCommentFlagValues.NONE

13.4.2.20 IgnorePassword

OIT Option ID: SCCOPT_IGNORE_PASSWORD

This option can disable the password verification of files where the contents can be processed without validation of the password. If this option is not set, the filter will return an error.

Currently only the PST and MDB Filters support this option.

Data Type

boolean

Default

false

13.4.2.21 ISODateTimes

OIT Option ID: SCCOPT_FORMATFLAGS

When this flag is set, all Date and Time values are converted to the ISO 8601 standard. This conversion can only be performed using dates that are stored as numeric data within the original file.

Data Type

boolean

Default
false

13.4.2.22 LotusNotesDirectory

OIT Option ID: SCCOPT_LOTUSNOTUSDIRECTORY

This option allows the developer to specify the location of a Lotus Notes or Domino installation for use by the NSF filter. A valid Lotus installation directory must contain the file nnotes.dll.

Either Lotus Notes version 8 or Lotus Domino version 8 must be installed on the same machine as OIT. A 32-bit version of the Lotus software must be used if you are using a 32-bit version of OIT. A 64-bit version of the Lotus software must be used if you are using a 64-bit version of OIT.

Data Type
String

Default
None

13.4.2.23 OutputRawtext

An option controlling whether raw text output is generated on the server side. This option must be enabled in order to enable in-browser searching via the Outside In API. This does not affect the search capabilities of the browser itself. Raw text generation is enabled by default.

Data Type
boolean

Default
true

13.4.2.24 PageRange

OIT Option ID: SCCOPT_WHATTOPRINT

OIT Option ID: SCCOPT_PRINTSTARTPAGE

OIT Option ID: SCCOPT_PRINTENDPAGE

This option indicates whether the whole file or a selected range of pages should be rendered. When selecting a range, the start and ending pages are specified.

Data Type
PageRange

Data
The page range to be exported.

Default
All pages are printed

13.4.2.25 PDFReorderBiDi

OIT Option ID: SCCOPT_PDF_FILTER_REORDER_BIDI

This option controls whether or not the PDF filter will attempt to reorder bidirectional text runs so that the output is in standard logical order as used by the Unicode 2.0 and later specification. This additional processing will result in slower filter performance according to the amount of bidirectional data in the file.

Data Type

PDFReorderBiDiValue

PDFReorderBiDiValue Enumeration

This enumeration defines the type of Bidirection text reordering the PDF filter should perform.

- STANDARDBIDI: Do not attempt to reorder bidirectional text runs.
- REORDEREDBIDI: Attempt to reorder bidirectional text runs.

Default

PDFReorderBiDiValue.STANDARDBIDI

13.4.2.26 PerformExtendedFI

OIT Option ID: SCCOPT_FIFLAGS

This option affects how an input file's internal format (application type) is identified when the file is first opened by the Outside In technology. When the extended test flag is in effect, and an input file is identified as being either 7-bit ASCII, EBCDIC, or Unicode, the file's contents will be interpreted as such by the export process.

The extended test is optional because it requires extra processing and cannot guarantee complete accuracy (which would require the inspection of every single byte in a file to eliminate false positives.)

Data Type

boolean

Data

One of the following values:

- false: When this is set, standard file identification behavior occurs.
- true: If set, the File Identification code will run an extended test on all files that are not identified.

Default

- true

13.4.2.27 PostLibraryScripts

List of strings specifying the relative URL of the customer's initialization script. Scripts specified via this option will load after the Outside In libraries, and may access the Outside In Javascript API. The scripts will be referenced in the same order as the list used to set this option.

Example:

```
<script src="oit.js">
<script src="/config/oitinit.js">
```

Data Type

List<String>

Default

None

13.4.2.28 PreLibraryScripts

List of strings specifying the URL of a script file to be referenced via a <script> tag in the output file. Scripts specified via this option will load before the Outside In libraries. The scripts will be referenced in the same order as the list used to set this option.

Example:

```
<script src="/jQuery/jQuery-min.js">
<script src="/shims/customstuff.js">
<script src="oit.js">
```

Data Type

List<String>

Default

None

13.4.2.29 ShowArchiveFullPath

This option causes the full path of a node to be returned in "GetRecordInfo" and "GetObjectInfo".

Data Type

boolean

Default

false

13.4.2.30 StrokeOutText

This option is used to stroke out (display as graphical primitives) text in an AutoCAD file. Setting this option to false would improve performance, but the visual fidelity may be compromised.

- If the export for the conversion is text only, text is never stroked out.
- If the export is not text only, and the drawing is perspective, text will always be stroked out (regardless of this option). This is due to the fact that in non-text only situations visual fidelity is of importance, and handling of textual objects in perspective drawings is more accurate with stroked out text. If the conversion is non-text only and the drawing is not perspective, this option determines if text should be stroked.

Note that when this option is true, some special characters appear as asterisks or question marks due to limited support of characters for stroking out text.

Data Type
boolean

Default
true

13.4.2.31 TimeZoneOffset
OIT Option ID: SCCOPT_TIMEZONE

This option allows the user to define an offset to GMT that will be applied during date formatting, allowing date values to be displayed in a selectable time zone. This option affects the formatting of numbers that have been defined as date values. This option will not affect dates that are stored as text.

> **Note:** This option does not apply for spreadsheet files. Daylight savings is not supported. The sent time in msg files when viewed in Outlook can be an hour different from the time sent when an image of the msg file is created.

Data Type
long

Default
- 0: GMT time

13.4.2.32 UnmappableCharacter
This option selects the character used when a character cannot be found in the output character set. This option takes the Unicode value for the replacement character. It is left to the user to make sure that the selected replacement character is available in the output character set.

Data Type
int

Default
42

13.4.2.33 URLPathOutput
String option for specifying the base URL for output files generated during the export process. The value of this option is used as a prefix for references to those resources in the primary output file. If the output base URL is "/exports/", an image tag in an exported document would look like this:

```
<img src="/exports/doc01.doc.jpg" />
```

and URLs to the exported files within content.js would be of this form:

```
 {id:"page.1",name:"Page
1",type:"page",oid:"overlay.1",w:"816px",h:"1056px",src:"/exports/output.3.html"}
```

By default, the output file base URL is empty, and all output will be specified without path information.

Data Type
String

Default
None

13.4.2.34 URLPathResources

String option for specifying the base URL for Web View Export scripts, css files and other resources that do not vary with the converted file. The value of this option is used as a prefix for references to those resources in the primary output file. For example, if the resources' base URL is "/oit/", the script reference in the output file would look like this:

```
<script src="/oit/oit.js"></script>
```

By default, this value is empty, meaning all resources must be present in the same directory as the generated output.

Data Type
String

Default
None

13.4.2.35 UseDocumentPageSettings

OIT Option ID: SCCOPT_USEDOCPAGESETTINGS

This option is used to select the document's page layout information when rendering.

If TRUE the document's native (or author selected) page margins, paper size, page scaling and page orientation are used when available from the filter.

The values of the DefaultPrintMargins, SSPrintGridlines, SSPrintHeadings, SSPrintDirection, and SSPrintFitToPage options are overridden if this option is set to TRUE and the properties associated with those options are specified in the input document. Additionally, print area and page breaks in spreadsheet documents are ignored unless this option is set to true.

If false, the page margins, size, orientation and scaling are set to specific values rather than those in the native document. The page size is forced to 8 1/2" x 11" in portrait orientation, but this may be changed by setting the GraphicHeight and/or GraphicWidth options. The margins are forced 1" all around, but may be changed by setting the defaultMargins option. The scaling for the document will be set to 100%, although this may be changed by setting any of the various scaling options.

It should be noted that this option also affects page orientation for both input spreadsheets and word processing documents.

Data Type
boolean

Default
true

13.4.2.36 WebFontPermissions

This option controls the use of "embedded" fonts (see FontReferenceMethod) in the Web View Export output; specifically, whether or not a downloadable (web font) version of the font should be made available to Web View Export output, based on the licensing restrictions indicated within the font itself.

Data Type

WebFontPermissionValues

WebFontPermissionValues Enumeration

WebFontPermissionValues can be one or more of the following enumerations ORed together:

- DEFAULTPERMISSIONS
- REQUIREINSTALLABLE
- REQUIREPREVIEWPRINT
- REQUIREEDITABLE

Default

WebFontPermissionValues.DEFAULTPERMISSIONS

13.4.2.37 WebViewLibraryName

String option specifying the file name of the Web View Javascript library, which will be written into a <script> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.js" will be used. This option allows the Web View Javascript library to be renamed for versioning or other purposes. This option works in combination with the URLPathResources option.

Data Type

String

Default

oit.js

13.4.2.38 WebViewOutputStructure

This option controls the structure of the output files created by Web View Export.

Data Type

WebViewStructureValue

WebViewStructureValue Enumeration

WebViewStructureValue can be one of the following enumerations:

- FLAT: HTML content is exported in a single HTML file.
- AJAXCHUNKED: HTML content is spread among multiple output pages.
- AJAXSTREAMED: HTML content is spread among multiple output pages and the browser may begin viewing the document before the entire output has been produced.

Default

WebViewStructureValue.FLAT

13.4.2.39 WebViewStylesheetName

String option specifying the file name of the Web View stylesheet, which will be written into a <link> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.css" will be used. This option allows the Web View stylesheet to be renamed for versioning or other purposes. This option works in combination with the URLPathResources option.

Data Type

String

Default

oit.css

13.5 ExportStatus Class

The ExportStatus class provides access to information about a conversion. This information may include information about sub-document failures, areas of a conversion that may not have high fidelity with the original document. When applicable the number of pages in the output is also provided.

Accessors

- long getPageCount() - A count of all of the output pages produced during an export operation.
- EnumSet<ExportStatusFlags> getStatusFlags() - Gets the information about possible fidelity issues with the original document.
- long getSubDocsFailed() - Number of sub documents that were not converted.
- long getSubDocsPassed() - Number of sub documents that were successfully converted.

ExportStatusFlags Enumeration

This enumeration is the set of possible known problems that can occur during an export process.

- NOINFORMATIONAVAILABLE: No Information is available.
- MISSINGMAP: A PDF text run was missing the toUnicode table.
- VERTICALTEXT: A vertical text run was present.
- TEXTEFFECTS: A run had unsupported text effects applied. One example is Word Art.
- UNSUPPORTEDCOMPRESSION: A graphic had an unsupported compression.
- UNSUPPORTEDCOLORSPACE: A graphic had an unsupported color space.
- FORMS: A subdocument had forms.
- RIGHTTOLEFTTABLES: A table had right to left columns.
- EQUATIONS: A file had equations.
- ALIASEDFONT: The desired font was missing, but a font alias was used.

- MISSINGFONT: The desired font wasn't present on the system.
- SUBDOCFAILED: A subdocument was not converted.

13.6 FileFormat Class

This class defines the identifiers for file formats.

Methods

- getDescription

  ```
  String getDescription()
  ```

  This method returns the description of the format.

- getId

  ```
  int getId()
  ```

  This method returns the numeric identifier of the format.

- forId

  ```
  FileFormat forId(int id)
  ```

  This method returns the FileFormat object for the given identifier.

  id: The numeric identifier for which the corresponding FileFormat object is returned.

13.7 FontAliases Class

FontAliases is a class for providing font matching of unknown fonts.

Constructor

```
FontAliases(boolean  clearDefaults, Map<String, String> aliasList)
   clearDefaults  Option to clear current list to defaults or empty list
   aliasList      Aliases list as a key-value pair with original name as key
```

Accessors

- Map<String, String> getAliasList() - List of font aliases set.
- boolean getClearDefaults() - When set to false, the existing alias list is cleared and set it to a list of default aliases. Otherwise the existing alias list is cleared out.

13.8 FontInfo Class

FontInfo is a class to define a font for use in the OutsideIn API.

Constructor

```
FontInfo(String fontface, int height)
   fontface  The name of the font
   height    Size of the font in half points
```

Accessors

- String getFontface() - The name of the font
- int getHeight() - Size of the font in half points

13.9 FontList Class

FontList is a class for inclusion or exclusion of fonts in exported documents.

Constructor

```
FontList(boolean  isExclusion, String[] fonts)
   IsExclusion  If set then accompanying list is an exclusion list
   fonts        List of fonts to include or exclude
```

Accessors

- boolean isExcludeList() - If set, then accompanying list is an exclusion list.
- String[] getFontsList - List of fonts to include or exclude.

13.10 HighlightAreaAnnotation Class

The HighlightAreaAnnotation class defines the characteristics of a highlighted area annotation. This class derives from the Annotation class.

Constructors

```
HighlightAreaAnnotation(long userId,  long sectionIx, long Top,
     long Left, long Width, long Height,  Annotation.UnitTypeValue units,
     float opacity, ColorInfo fillColor,
     HighlightAreaAnnotation.AnnoBorder border,
     HighlightAreaAnnotation.BorderStyleValue borderStyle
userId       User Data
sectionIx    0-based page/sheet/image/slide index
Top          Top coordinate or row
Left         Leftmost coordinate or column
Width        Width of area in coordinates or columns
Height       Height of area in coordinates or rows
Units        Unit type
Opacity      Opacity of the annotation. Range 0.0 - 1.0; 0==invisible
fillColor    The fill color for the annotation
border       The color and thickness of the borders for the highlighted area.
borderStyle  Style of border lines surrounding the highlighted area.
```

Accessors

- **BorderStyle**: The style of the border. (Note: in the initial release of Web View Export, only solid borders are supported)

  ```
  void setBorderStyle(HighlightAreaAnnotation.BorderStyleValue)
  HighlightAreaAnnotation.BorderStyleValue getBorderStyle()
  ```

- **FillColor**: Color to be used to fill the area

  ```
  void setFillColor(ColorInfo)
  ColorInfo getFillColor()
  ```

- **BottomBorder**

  ```
  void setBottomBorder(HighlightAreaAnnotation.AnnoBorder)
  HighlightAreaAnnotation.AnnoBorder getBottomBorder()
  ```

- **LeftBorder**

  ```
  void setLeftBorder(HighlightAreaAnnotation.AnnoBorder)
  HighlightAreaAnnotation.AnnoBorder getLeftBorder()
  ```

- **RightBorder**

```
void setRightBorder(HighlightAreaAnnotation.AnnoBorder)
HighlightAreaAnnotation.AnnoBorder getRightBorder()
```

- **TopBorder**

```
void TopBorder(HighlightAreaAnnotation.AnnoBorder)
HighlightAreaAnnotation.AnnoBorder getTopBorder()
```

Methods

```
void setBorder(HighlightAreaAnnotation.BorderSideValue side,
HighlightAreaAnnotation.AnnoBorder)
```

side: The side to which this border shall be applied. HighlightAreaAnnotation.BorderSideValue is an enumeration of the sides of the annotation.

border: The type of border to be used

13.10.1 AnnoBorder Class

The AnnoBorder class defines the characteristics of the borders of annotations.

Constructors

```
AnnoBorder(ColorInfo color, long thickness)
   color: Color of the annotation border
   thickness: Thickness of the annotation border
```

Initializes a new instance of the AnnoBorder class.

13.11 HighlightTextAnnotation Class

HighlightTextAnnotation is a class for defining Text highlighted Annotations. This class derives from the Annotation class.

> **Note:** the application of character attributes through highlighting is not supported in the initial release of Web View Export.

Constructors

```
HighlightTextAnnotation(long StartCharCount,
                long  EndCharCount,
                EnumSet<CharAttributeValues> CharAttrs,
                Enumset<CharAttributeValues> CharMask)
HighlightTextAnnotation(long StartCharCount,
                long EndCharCount,
                ColorInfo Foreground,
                ColorInfo Background)
HighlightTextAnnotation(long StartCharCount,
                long EndCharCount,
                ColorInfo Foreground,
                ColorInfo Background,
                EnumSet<CharAttributeValues> CharAttrs,
                EnumSet<CharAttributeValues> CharMask)
StartCharCount  The character count of the starting position
EndCharCount    The character count of the end position
Foreground      The text color of the highlight
```

```
Background      The background color of the highlight
CharAttrs       the character attributes to use
CharMask        character attributes to change
```

Initializes a new instance of the HighlightTextAnnotation class.

Accessors

- **StartCharCount**: The character count of the starting position
- **EndCharCount**: The character count of the end position
- **Foreground**: The text color of the highlight
- **Background**: The background color of the highlight
- **CharAttrs**: The character attributes to use
- **CharMask**: Character attributes to change

CharAttributeValues Enumeration

This enumeration is the list of all character attributes to apply for the text highlight.

- NORMAL: Normal text - All attributes off
- UNDERLINE: Underline attribute
- ITALIC: Italic attribute
- BOLD: Bold attribute
- STRIKEOUT: Strike out text
- SMALLCAPS: Small caps
- OUTLINE: Outline Text
- SHADOW: Shadow text
- CAPS: All Caps
- SUBSCRIPT: Subscript text
- SUPERSCRIPT: Superscript text
- DUNDERLINE: Double Underline
- WORDUNDERLINE: Word Underline
- DOTUNDERLINE: Dotted Underline
- DASHUNDERLINE: Dashed Underline

13.12 Margins Class

The Margins Class is used to describe the page margins.

Constructor

```
Margins(long top,
       long bottom,
       long left,
       long right)
top    Margin from the top edge of the page (in twips)
bottom Margin from the bottom edge of the page (in twips)
left   Margin from the left edge of the page (in twips)
right  Margin from the right edge of the page (in twips)
```

Accessors

- long getTop() - Margin from the top edge of the page (in twips)
- long getBottom() - Margin from the bottom edge of the page (in twips)
- long getLeft() - Margin from the left edge of the page (in twips)
- long getRight() - Margin from the right edge of the page (in twips)

13.13 OutsideIn

This is a utility class that creates an instance of an Exporter object on request.

13.14 OutsideInException Class

This is the exception that is thrown when an Outside In Technology error occurs.

This class derives from the Exception class. This class has no public methods or properties except those of the parent Exception class.

13.15 PageInfo Class

PageInfo is a class for defining page dimensions.

Constructor

```
PageInfo(PageInfo.PageSizeUnitsValue units,
              float width,
              float height)
units    Units used to specify width and height
width    Width of the page
height   Height of the page
```

Accessors

- PageInfo.PageSizeUnitsValue getUnits() - Units used to specify width and height
- float getWidth() - Width of the page
- float getHeight() - Height of the page

PageInfo.PageSizeUnitsValue Enumeration

PageSizeUnitsValue is an enumeration of the various units that can be used to specify the width and height of a page.

- INCHES: Units are in Inches
- POINTS: Units are in Points (1/72th of an inch)
- CENTIMETERS: Units are in Centimeters
- PICAS: Units are in Picas (1/6th of an inch)

13.16 PageRange Class

PageRange is a class for defining page ranges for exporting purposes.

Constructors

```
PageRange()
```

Creates an instance of the PageRangeObject for printing all pages.

```
PageRange(long StartPage)
StartPage  Starting page number of the print range
```

Creates an instance of the PageRangeObject for printing from a page until end of document.

```
PageRange(
          long  StartPage,
          long  StopPage) throws OutsideInException
StartPage  Starting page number of the print range
StopPage   End page number of the print range
```

Creates an instance of the PageRangeObject for printing a range of pages.

Accessors

- boolean getPrintAll() - If set to true, all pages of the document will be printed
- long getStartPage() - The start page of the print range. 0 indicates printing will begin with the first page of the document.
- long getStopPage() - The last page of the print range. 0 indicates the last page at the end of the document.

13.17 StampAnnotation Class

StampAnnotation is the class for defining stamp annotations. This class derives from the Annotation class.

The StampAnnotation type exposes the following members.

Constructors

```
StampAnnotation(long userId,
   long sectionIx,
   long Top,
   long Left,
   long Width,
   long Height,
   Annotation.UnitTypeValue units,
   float opacity,
   StampAnnotation.SizingMethodValue sizemode,
   String stampName)
userId     User Data
sectionIx  0-based page/sheet/image/slide index
Top        Top coordinate or row
Left       Leftmost coordinate or column
Width      Width of area in coordinates or columns
Height     Height of area in coordinates or rows
Units      Unit type
Opacity    Opacity of the annotation. Range 0.0 - 1.0; 0==invisible
Sizemode   Image sizing method
stampName  Name of the stamp
```

Accessors

SizeMode: Sizing method of the source image

```
void setSizeMode(StampAnnotation.SizingMethodValue)
StampAnnotation.SizingMethodValue getSizeMode()
```

StampName: Registered name of stamp image

```
void setStampName(String)
String getStampName()
```

SizingMethodValue Enumeration

The SizingMethodValue is an enumeration of the sizing methods that can be applied to the image.

13.18 TreeRecord Class

TreeRecord provides information about an archive record. This is a read-only class where the technology fills in all the values.

Accessors

- boolean isFolder() - A value of true indicates that the record is a directory node.
- int getFileSize() - File size of the tree record
- int getTime() - Time the tree record was created
- int getNodeNum() - Serial number of the tree record in the archive
- String getNodeName() - The name of the tree record

Part V

Using the .NET API

This section provides details about using the Web View Export SDK with the .NET API.

Part V contains the following chapters:

- Chapter 14, "Web View Export .NET API"
- Chapter 15, "Web View Export .NET Classes"

14

Web View Export .NET API

Outside In .NET is a set of class libraries and Windows DLLs that provides developers an easy interface to create .NET applications using Outside In Technology.

14.1 Requirements

To develop applications using the .NET APIs, the following set of modules and tools are required:

- The Outside In Technology (OIT) developer's redistributable modules for your product
- Visual Studio 2010 or later
- NET Framework 4.0 or later
- The API libraries:

  outsidein.dll - The .NET libraries to access the Outside In technologies

  oilink.dll and oilink.exe- The bridge modules between .NET and the C-APIs

  Google.ProtocolBuffers.dll - The cross language binary serialization provider

14.2 Getting Started

There are two steps in developing applications using the APIs. In the first step, you would need to configure the environment to create your application (typical programming tasks not directly related to these APIs) and in the second step you would generate code to utilize the functionality of these libraries.

14.2.1 Configuring your Environment

To set up the environment to create a .NET application, you would need to add a reference to the Outside In library. In order to use the Outside In components in your application, you need to reference outsidein.dll. (This can be done by using the Add Reference dialog box in Visual Studio.)

14.2.2 Generate Code

The sample application included with the SDK, wv_sample_exporter_cs, is a minimal demonstration of how to use this API.

All the functionality required to perform a conversion is provided in an Exporter object. The basic process of exporting a file involves the following tasks:

1. Create an Exporter object. To obtain access to the Outside In functionality, you should call the utility function in the "OutsideIn" class. This will provide you an instance of an Exporter Object.
2. Configure the export. The Outside In API is highly configurable, and presents numerous options to fine-tune the way a document is exported. Each option has a "set" and "get" method to set or retrieve the currently set value.
3. Set the source and primary destination files. You are required to specify the source file and the destination file. This is done similar to setting options using "set" methods.
4. Set the output type. In this step, you specify the output format.
5. (Optional) Provide a callback handler. The Outside In Technology provides callbacks that allow the developer to intervene at critical points in the export process. To respond to these callbacks, you would have to subscribe to any messages that you are interested in by overriding the message handlers from the "Callback" class. After creating an object of this class, set the callback option to this object and the messages will be passed to your object.
6. Run the export. After all the previous steps are completed, you can produce the desired output.

14.2.2.1 Create an Exporter Object

To obtain access to the Outside In functionality, you should call the utility function in the "OutsideIn" class. This will provide you an instance of an Exporter Object.

```
Exporter exporter = OutsideIn.OutsideIn.NewLocalExporter();
```

If the Outside In binaries (including oilink.exe) are located in a directory other than the one from which outsidein.dll was loaded, the OutsideIn class may need to be told where to find them. (Otherwise you may see an OutsideInException indicating that oilink.exe could not be found.) You can specify the location of the binaries by setting the static property OutsideIn.InstallLocation to the path of the directory where they are installed. This property should only be set once. This property is only available in the .NET APIs in version 8.5.0.1 and beyond.

14.2.2.2 Configure the Output

The Outside In API is highly configurable, and presents numerous options to fine-tune the way a document is exported. Each option has a "set" and "get" method to set or retrieve the currently set value.

```
exporter.SetPerformExtendedFI(true);
int timezoneOffset = exporter.GetTimeZoneOffset();
```

14.2.2.3 Set the Source and Primary Destination Files

You are required to specify the source file and the destination file. This is done similarly to setting options using "set" methods.

```
exporter.SetSourceFile(inputFilename);

exporter.SetDestinationFile(outputFilename);
```

There are other options that can be set at this time to specify the way to handle the input file, such as providing a SourceFormat to provide a mechanism to handle the input file in a different format than that which it is identified as.

The API also supports opening certain types of embedded documents from within an input file. For example, a .zip file may contain a number of embedded documents; and an email message saved as a .msg file may contain attachments. The API provides the means of opening these types of embedded documents. This can be done by opening the parent document and then the embedded document can be opened through this exporter object.

```
// subdocId is the sequential number of the node in the archive file

Exporter exporterNode = exporter.NewTreeNodeExporter(subdocId);
```

14.2.2.4 Set the Output Type

In this step, you specify the output format.

```
exporter.SetDestinationFormat(FileFormat.FI_HTML5);
```

14.2.2.5 Provide a Callback Handler

Outside In Technology provides callbacks that allow the developer to intervene at critical points in the export process. To respond to these callbacks, you have to subscribe to any messages that you are interested in by overriding the message handlers from the Callback class. After creating an object of this class, set the callback option to this object and the messages will be passed to your object.

```
class CallbackHandler : Callback

{

  … // implementation of messages to handle - described in the API documentation

}

CallbackHandler callback = new CallbackHandler();

exporter.SetCallbackHandler(callback);
```

14.2.2.6 Run the Export

After all the previous steps are completed, you can produce the desired output.

```
exporter.Export();
```

15

Web View Export .NET Classes

This section describes the Web View Export .NET classes.

15.1 Annotation Class

Annotation is an abstract base class for the Annotation objects.

Properties

- Height (Int64) Height of area in coordinates or rows
- Left (Int64) Leftmost coordinate or column
- Opacity (Single) Opacity of the annotation. Range 0.0 - 1.0; 0==invisible
- SectionIndex (Int64) 0-based page/sheet/image/slide index
- Top (Int64) Top coordinate or row
- Units (Annotation.UnitTypeValue) Type of units in which Height, Width, Left and Top are described
- UserId (Int64) User Data
- Width (Int64) Width of area in coordinates or columns

Annotation.UnitTypeValue Enumeration

The UnitTypeValue is an enumeration of the various unit types that annotation positions can be described in.

- Pixels: Units specified in Pixels
- Twips: Units specified in Twips (1/1440th of an inch)
- Cells: Units specified in cell positions

15.2 Callback Object

Callback messages are notifications that come from Outside In during the export process, providing information and sometimes the opportunity to customize the generated output. To access callback messages, your code must create an object that inherits from Callback and pass it through the API's SetCallbackHandler method. Your object can implement methods that override the default behavior for whichever methods your application is interested in.

Callback has two methods: CreateNewFile and NewFileInfo.

15.2.1 CreateNewFile

```
CreateNewFileResponse CreateNewFile( FileFormat ParentOutputId,  FileFormat
OutputId,
   Association Association, string Path)
```

This callback is made any time a new output file needs to be generated. This gives the developer the chance to affect where the new output file is created, how it is named, and the URL (if any) used to reference the file.

Parameters

- ParentOutputId: File format identifier of the parent file.
- OutputId: File format identifier of the file created.
- Association: An Association that describes relationship between the primary output file and the new file.
- Path: Full path of the file to be created.

Return Value

To take action in response to this notification, return a CreateNewFileResponse object with the new file information. If you wish to accept the defaults for the path and URL, you may return null.

CreateNewFileResponse Class

This is a class to define a new output file location in response to a CreateNewFile callback. If you do not wish to change the path to the new output file, you may use the path as received. If you do not wish to specify the URL for the new file, you may specify it as null.

Constructor

```
CreateNewFileResponse(FileInfo File, string URL)
```

- File: File object with full path to new file.
- URL: A new URL that references the newly created file. This parameter can be null.

Association Enumeration

This enumeration defines, for a new file created by an export process, the different possible associations between the new file and the primary output file. Its value may be one of the following:

- Root - indicates the primary output file
- Child - a new file linked (directly or indirectly) from the primary output file
- Sibling - indicates new files not linked from the primary output file
- Copy - the file was copied as a part of a template macro operation.
- RequiredName - not used

Note that some of these relationships will not be possible in all Outside In Export SDKs.

15.2.2 NewFileInfo

```
void NewFileInfo( FileFormat ParentOutputId, FileFormat OutputId,
```

```
Association Association, string Path, string URL)
```

This informational callback is made just after each new file has been created.

Parameters

- ParentOutputId: File format identifier of the parent file
- OutputId: File format identifier of the file created
- Association: An Association that describes relationship between the primary output file and the new file.
- Path: Full path of the file created
- URL: URL that references the newly created file

15.3 ColorInfo Class

ColorInfo is a class to define a color or to use a default color in appropriate cases.

Constructors

```
ColorInfo()
```

Initializes a ColorInfo object to use the default color.

```
public ColorInfo(Byte red,
   Byte green,
   Byte blue)
```

Initializes a ColorInfo object with the specified RGB values.

Properties

- Blue (byte) - Blue component of the color
- Green (byte) - Green component of the color
- Red (byte) - Red component of the color
- UseDefault (byte) - Set to true if the default color is used

15.4 Exporter Interface

This section describes the properties and methods of Exporter.

All of Outside In's Exporter functionality can be accessed through the Exporter Interface. The object returned by OutsideIn class is an implementation of this interface. This class derives from the Document Interface, which in turn is derived from the OptionsCache Interface.

Methods

- **AddKeyValuePairs**

  ```
  void AddKeyValuePairs(
    Dictionary<string, string> stringkvPairs)
  ```

  This function lets you specify key/value pairs for use in the exported document.

  stringkvPairs: Key value pairs where the key is the name of the data.

- **AddStampAnnotation**

```
void AddStampAnnotation(
     StampAnnotation StampAnno
)
```

Add a stamp annotation to the exported document.

StampAnno: A StampAnnotation object with information about the stamp to add.

- **AddStampAnnotation and Add Annotation Properties**

```
void AddStampAnnotation(
      StampAnnotation StampAnno,
      Dictionary<string, string> Properties)
```

Add a stamp annotation to the exported document and associate properties with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add.

Properties: Key value pairs of name/value of properties associated with this annotation.

- **AddStampAnnotation and Associate Comment with Annotation**

```
void AddStampAnnotation(
      StampAnnotation StampAnno,
      string Comment)
```

Add a stamp annotation to the exported document and associate a comment with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add

Comment: Comment text to associate with the annotation

- **AddStampAnnotation with Comment and Properties**

```
void AddStampAnnotation(
      StampAnnotation StampAnno,
      string Comment,
      Dictionary<string, string> Properties)
```

Add a stamp annotation to the exported document and associate a comment and properties with the annotation.

StampAnno: A StampAnnotation object with information about the stamp to add

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **AddStampImage**

```
void AddStampImage(
      Dictionary<string, FileInfo> stamps
)
```

This method maps a stamp name to an image file.

stamps: Key value pairs of stamp names and image files

- **AddStampImage**

```
void AddStampImageURL(
      Dictionary<string, Uri> stamps
)
```

This method maps a stamp name to an image URL.

stamps: Key value pairs of stamp names and image URLs

- **ApplyHighlights**

```
void ApplyHighlights(
      string HighlightJsonData
)
```

This method applies a set of highlights from a JSON-encoded text stream previously generated from the Web View Javascript library.

HighlightJsonData: stream of highlight (and comment) information that was obtained via the Web View Javascript API function OIT.highlights.serialize.

- **GetExportStatus**

```
ExportStatus GetExportStatus()
```

This function is used to determine if there were conversion problems during an export. The ExportStatus object returned may have information about sub-document failures, areas of a conversion that may not have high fidelity with the original document. When applicable the number of pages in the output is also provided.

- **AddAreaHighlight**

```
void AddAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno
)
```

Highlight an area on a page in a document.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

- **AddAreaHighlight and Add Annotation Properties**

```
void AddAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      Dictionary<string, string> Properties
)
```

Add an area highlight to a document and associate properties with the annotation.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Properties: Key value pairs of name/value of properties associated with this annotation

- **AddAreaHighlight and Associate Comment with Annotation**

```
void AddAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      string Comment
)
```

Highlight an area on a page in a document and associate a comment with the annotation.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Comment: Comment text to associate with the annotation

- **AddAreaHighlight with Comment and Properties**

```
void AddAreaHighlight(
      HighlightAreaAnnotation HighlightAreaAnno,
      string Comment,
      Dictionary<string, string> Properties
)
```

Add an area highlight to a document, including associated comment text and properties.

HighlightAreaAnno: A HighlightAreaAnnotation object with information about the area to highlight

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **AddTextHighlight**

```
void AddTextHighlight(
      HighlightTextAnnotation textanno
)
```

Highlight text in a document.

textanno: A HighlightTextAnnotation object with information about the text to highlight

- **AddTextHighlight and Add Annotation Properties**

```
void AddTextHighlight(
      HighlightTextAnnotation textanno,
      Dictionary<string, string> Properties
)
```

Highlight text in a document and associate properties with the annotation.

textanno: A HighlightTextAnnotation object with information about the text to highlight

Properties: Key value pairs of name/value of properties associated with this annotation

- **AddTextHighlight and Associate a Comment**

```
void AddTextHighlight(
      HighlightTextAnnotation textanno,
      string Comment
)
```

Highlight text in a document and associate a comment with the highlight.

textanno: A HighlightTextAnnotation object with information about the text to highlight

Comment: Comment text to associate with the annotation

- **AddTextHighlight with Comment and Properties to Annotation**

```
void AddTextHighlight(
      HighlightTextAnnotation textanno,
      string Comment,
```

```
      Dictionary<string, string> Properties
)
```

Highlight text in a document and provide comment text and properties to be associated with the annotation.

textanno: An HighlightTextAnnotation object with information about the text to highlight

Comment: Comment text to associate with the annotation

Properties: Key value pairs of name/value of properties associated with this annotation

- **NewSubDocumentExporter**

```
Exporter NewSubDocumentExporter(
      int SubdocId,
      SubDocumentIdentifierTypeValue idType
)
```

Create a new Exporter for a subdocument.

SubDocId: Identifier of the subdocument

idType: Type of subdocument

SubDocumentIdentifierTypeValue: This is an enumeration for the type of subdocument being opened.

- XMLExportLocator: Subdocument to be opened is based on output of XML Export (SubdocId is the value of the object_id attribute of a locator element.)
- AttachmentLocator: Subdocument to be opened is based on the locator value provided by the one of the Export SDKs.
- EmailAttachmentIndex: Subdocument to be opened is based on the index of the attachment from an email message. (SubdocId is the zero-based index of the attachment from an email message file. The first attachment presented by OutsideIn has the index value 0, the second has the index value 1, etc.)

Returns: A new Exporter object for the subdocument

- **NewTreeNodeExporter**

```
Exporter NewTreeNodeExporter(
      int dwRecordNum
)
```

Create a new Exporter for an archive node. You may get the number of nodes in an archive using getArchiveNodeCount. The nodes are numbered from 0 to getArchiveNodeCount -1.

dwRecordNum: The number of the record to retrieve information about. The first node is node 0 and the total number of nodes may be obtained from GetArchiveNodeCount.

Returns: A new Exporter object for the archive node

- **NewTreeNodeExporter with Search Export Data**

```
Exporter NewTreeNodeExporter(
      uint flags,
      uint params1,
      uint params2
)
```

Create a new Exporter for an archive node. To use this function, you must first process the archive with Search Export and save the Node data for later use in this function.

Flags: Special flags value from Search Export

Params1: Data1 from Search Export

Params2: Data2 from Search Export

Returns: A new Exporter object for the archive node

- **Export**

```
void Export()
```

Perform the conversion.

- **SetDestinationFile**

```
OptionsCache SetDestinationFile(
      string filename
)
```

Set the location of the destination file.

filename: Full path to the destination file

returns: The updated options object

15.4.1 Document Interface

All of the Outside In document-related methods are accessed through the Document Interface.

Methods

- **Close**

```
void Close()
```

Closes the currently open document.

- **GetArchiveNodeCount**

```
Int32 GetArchiveNodeCount()
```

Retrieves the number of nodes in an archive file.

Returns the number of nodes in the archive file or 0 if the file is not an archive file.

- **GetFileId**

```
FileFormat GetFileId(FileIdInfoFlagValue dwFlags)
```

Gets the format of the file based on the technology's content-based file identification process.

dwFlags: Option to retrieve the file identification pre-Extended or post-Extended Test

Returns the format identifier of the file.

- **GetObjectInfo**

```
ObjectInfo GetObjectInfo()
```

Retrieves the information about an embedded object.

Return: An ObjectInfo object with the information about the embedded object

- **GetRecordInfo**

```
TreeRecord GetRecordInfo(Int32 nNodeNum)
```

Retrieves information about a record in an archive file

nNodeNum: The number of the record to retrieve information about. The first node is node 0.

Return Value: A TreeRecord object with the information about the record

- **SaveRecord**

```
void SaveRecord(
      Int32 nNodeNum,
      FileInfo fileinfo)
void SaveRecord(
      Int32 nNodeNum,
      string strFileName)
```

Extracts a record in an archive file to disk.

nNodeNumType: The number of the record to retrieve information about. The first node is node 0.

strFileNameType/fileinfo: Full path of the destination file to which the file will be extracted

- **SaveRecord with Tree Record**

```
void SaveRecord(
      TreeRecord treeRec,
      FileInfo fileinfo)
void SaveRecord(
      TreeRecord treeRec,
      string strFileName)
```

Extracts a record in an archive file to disk.

treeRec: A TreeRecord object retrieved from GetRecordInfo with information about the node to extract

strFileNameType/fileinfo: Full path of the destination file to which the file will be extracted

- **SetSourceFile**

```
OptionsCache SetSourceFile( string filename)
```

Set the source document.

filename: Full path of the source document

Returns: The options cache object associated with this document

15.4.2 OptionsCache Class

This section describes the OptionsCache class.

The options that configure the way outputs are generated are accessed through the OptionsCache class.

All of the options described in the following subsections are available through this interface. Other methods in this interface are described below.

Methods

- `OptionsCache SetSourceFile(FileInfo file)`

  Sets the source document to be opened.

  file: Full path to source file

- `OptionsCache SetSourceFormat(FileFormat fileId)`

  Sets the source format to process the input file as, ignoring the algorithmic detection of the file type.

  fileId: the format to treat the input document as.

- `OptionsCache SetDestinationFile(FileInfo file)`

  Sets the location of the destination file.

  file: Full path to the destination file

- `OptionsCache SetDestinationFormat(FileFormat fileId)`

  Sets the destination file format to which the file should be converted to.

  fileId: the format to convert the input document(s) to.

- `OptionsCache SetCallbackHandler(Callback callback)`

  Sets the object to use to handle callbacks.

  callback: the callback handling object.

- `OptionsCache SetPasswordsList(List<String> Passwords)`

  Provides a list of strings to use as passwords for encrypted documents. The technology will cycle through this list until a successful password is found or the list is exhausted.

  Passwords: List of strings to be used as passwords.

- `OptionsCache SetLotusNotesId(String NotesIdFile)`

  Sets the Lotus Notes ID file location.

  NotesIdFile: Full path to the Notes ID file.

15.4.2.1 BiDiReorderMethod

OIT Option ID: SCCOPT_REORDERMETHOD

This option controls how the technology reorders bidirectional text.

Data Type

- BiDiReorderMethodValue

BiDiReorderMethodValue Enumeration

One of the following values:

- UnicodeOff: This disables any processing for bidirectional characters. This option is the default.

- UnicodeLToR: Characters displayed using the Unicode bidirectional algorithm assuming a base left-to-right order. Use this option to enable bidirectional rendering.
- UnicodeRToL: Characters displayed using the Unicode bidirectional algorithm assuming a base right-to-left order. Use this option to force starting bidirectional rendering in the right-to-left.

Default

BiDiReorderMethodValue.UnicodeOff

15.4.2.2 DefaultInputCharacterSet

OIT Option ID: SCCOPT_DEFAULTINPUTCHARSET

This option is used in cases where Outside In cannot determine the character set used to encode the text of an input file. When all other means of determining the file's character set are exhausted, Outside In will assume that an input document is encoded in the character set specified by this option. This is most often used when reading plain-text files, but may also be used when reading HTML or PDF files.

Data Type

DefaultInputCharacterSetValue

DefaultInputCharacterSetValue Enumeration

DefaultInputCharacterSetValue can be one of the following enumerations:

SystemDefault

Unicode

BigEndianUnicode

LittleEndianUnicode

Utf8

Utf7

Ascii

UnixJapanese

UnixJapaneseEuc

UnixChineseTrad1

UnixChineseEucTrad1

UnixChineseTrad2

UnixChineseEucTrad2

UnixKorean

UnixChineseSimple

Ebcdic37

Ebcdic273

Ebcdic274

Ebcdic277

Ebcdic278

Ebcdic280

Ebcdic282

Ebcdic284

Ebcdic285

Ebcdic297

Ebcdic500

Ebcdic1026

Dos437

Dos737

Dos850

Dos852

Dos855

Dos857

Dos860

Dos861

Dos863

Dos865

Dos866

Dos869

Windows874

Windows932

Windows936

Windows949

Windows950

Windows1250

Windows1251

Windows1252

Windows1253

Windows1254

Windows1255

Windows1256

Windows1257

Iso8859_1

Iso8859_2

Iso8859_3

Iso8859_4

Iso8859_5

Iso8859_6

Iso8859_7

Iso8859_8

Iso8859_9

MacRoman

MacCroatian

MacRomanian

MacTurkish

MacIcelandic

MacCyrillic

MacGreek

MacCE

MacHebrew

MacArabic

MacJapanese

HPRoman8

BiDiOldCode

BiDiPC8

BiDiE0

RussianKOI8

JapaneseX0201

Default

DefaultInputCharacterSetValue.SystemDefault

15.4.2.3 DefaultPageMargins

This option specifies the top, left, bottom and right margins in twips from the edges of the page. For instance, setting all the values to 1440 creates a 1-inch margin on all sides. Page margins will only be applied when formatting word processing, database and spreadsheet files.

Please note all margins are applied before scaling with the PageFitMode option. This option is overridden when the UseDocumentPageSettings option is set to true and print margins are specified in the input document. This option does not affect the output of bitmap, presentation, vector or archive files.

Data Type

Margins

Data

A Margins object with the margins on the 4 sides defined.

Default

1 inch for all margins

15.4.2.4 DefaultPageSize

This option allows the developer to specify the size of each page in the generated output file. The size may be specified in inches, points, centimeters or picas. This option is only valid when UseDocumentPageSettings is set to false. 1 inch = 6 picas = 72 points = ~ 2.54 cm.

Data Type

PageInfo

Data

A PageInfo object with the page size information.

Default

8.5 inches by 11 inches

15.4.2.5 DefaultRenderFont

This option sets the font to use when the font specified by the filter is not available on the system.

Data Type

FontInfo

Data

The default font to be used

Default

None

15.4.2.6 DocumentMemoryMode

OIT Option ID: SCCOPT_DOCUMENTMEMORYMODE

This option determines the maximum amount of memory that the internal storage may use to store the document's data, from 4 MB to 1 GB. The more memory the internal storage has available to it, the less often it needs to re-read data from the document.

Data

- Smallest: 1 - 4MB
- Small: 2 - 16MB
- Medium: 3 - 64MB
- Large: 4 - 256MB
- Largest: 5 - 1 GB

Default

DocumentMemoryModeValue.Small

15.4.2.7 EMailAttachmentHandling

This option determines whether email attachments are converted or extracted.

Data Type

EmailAttachmentHandlingValue

EmailAttachmentHandlingValue Enumeration

- None: Attachments will not be converted or extracted.
- Export: The attachments to this email will be converted.
- ExportRecursive: The attachments to this email will be converted; if any are emails, they will also have their attachments converted, and so on.
- Extract: Attachments will be extracted in their native form, stored alongside the Web View output files.

Default

EmailAttachmentHandlingValue.None

15.4.2.8 ExternalStylesheets

String option for specifying the url of a user supplied stylesheet to be included in the output html file. Values specified through this option will appear in output HTML files as linked stylesheets.

Example

```
<script src="oit.css">
<link rel="stylesheet" href="/some/path/myStylesheet.css"/>
```

Data Type

List<string>

Default

None

15.4.2.9 FallbackFormat

This option controls how files are handled when their specific application type cannot be determined. This normally affects all plain-text files, because plain-text files are generally identified by process of elimination, for example, when a file isn't identified as having been created by a known application, it is treated as a plain-text file. It is recommended that None be set to prevent the conversion from exporting unidentified binary files as though they were text, which could generate many pages of "garbage" output.

Data Type

FallbackFormatValue

FallbackFormatValue Enumeration

FallbackFormatValue can be one of the following enumerations:

- Text: Unidentified file types will be treated as text files.
- None: Outside In will not attempt to process files whose type cannot be identified.

Default

FallbackFormatValue.Text

15.4.2.10 FontAliasList

This option enables the capability to specify which font on the system should be used when a specific font referenced in the original file is not available. A different alias can be set for each font desired to be mapped.

Data Type

FontAliases

Data

A FontAliases object with a list of font matchings

Default

Please refer to the section "SCCOPT_PRINTFONTALIAS" in the C-API for the default font alias list.

15.4.2.11 FontBaseURL

This option allows the developer to specify the base URL used in the Web View Export output when referencing a non-exported font as a downloadable font. (Exported fonts will be placed in the exported output location.)

Data Type

String

Default

None

15.4.2.12 FontDirectories

This option allows the developer to specify one or more font directories where fonts are located for use by Web View Export. If multiple font directories are specified, they should be delimited by a colon on Linux and UNIX systems and a semi-colon on Windows systems. Please note that Web View Export supports single TrueType fonts (*.ttf, *.TTF) and TrueType collections (*.ttc, *.TTC), not Windows bitmap fonts (*.fon, *.FON), or any other type of font. Also, Web View Export does not require case-sensitive font filenames on UNIX systems.

Data Type

List<DirectoryInfo>

Default

None

15.4.2.13 FontFilter

This option allows the developer to specify a list of fonts to be included or excluded during the export process

Data Type

FontList

Data

A FontFilter object describing the inclusion or exclusion list.

Default
None

15.4.2.14 FontReferenceMethod
This option controls the way embedded fonts are presented (or not) in the HTML5 output.

Data Type
FontReferenceMethodValue

FontReferenceMethodValue Enumeration
- ReferenceByName: The font is referenced in the output file by name only
- ReferenceExported: The font is subsetted, exported during document conversion, and referenced in the output by file name
- ReferenceByBaseURL: The full font is referenced in the output using the font file name and the base URL supplied by the FontBaseURL option.

Default
FontReferenceMethodValue.ReferenceByName

15.4.2.15 GraphicHeightLimit
OIT Option ID: SCCOPT_GRAPHIC_HEIGHTLIMIT

Note that this option differs from the behavior of setting the height of graphics in that it sets an upper limit on the image height. Images larger than this limit will be reduced to the limit value. However, images smaller than this height will not be enlarged when using this option. Setting the height using GraphicHeight causes all output images to be reduced or enlarged to be of the specified height.

Data Type
UInt32

Default
0

15.4.2.16 GraphicOutputDPI
OIT Option ID: SCCOPT_GRAPHIC_OUTPUTDPI

This option allows the user to specify the output graphics device's resolution in DPI and only applies to images whose size is specified in physical units (in/cm). For example, consider a 1" square, 100 DPI graphic that is to be rendered on a 50 DPI device (GraphicOutputDPI is set to 50). In this case, the size of the resulting TIFF, BMP, JPEG, GIF, or PNG will be 50 x 50 pixels.

In addition, setting this option to 0 (Maintain Source Image DPI) can be used to suppress any dimensional changes to an image. In other words, a 1" square, 100 DPI graphic will be converted to an image that is 100 x 100 pixels in size. This value indicates that the DPI of the output device is not important. It extracts the maximum resolution from the input image with the smallest exported image size.

Setting this option to 0 may result in the creation of extremely large images. Be aware that there may be limitations in the system running this technology that could result in undesirably large bandwidth consumption or an error message. Additionally, an out of

memory error message will be generated if system memory is insufficient to handle a particularly large image.

Also note that setting this option to 0 will force the technology to use the DPI settings already present in raster images, but will use the current screen resolution as the DPI setting for any other type of input file.

For some output graphic types, there may be a discrepancy between the value set by this option and the DPI value reported by some graphics applications. The discrepancy occurs when the output format uses metric units (DPM, or dots per meter) instead of English units (DPI, or dots per inch). Depending on how the graphics application performs rounding on meters to inches conversions, the DPI value reported may be 1 unit more than expected. An example of a format which may exhibit this problem is PNG. The maximum value that can be set is 2400 DPI.

Data Type

UInt32

Default

96

15.4.2.17 GraphicSizeLimit

OIT Option ID: SCCOPT_GRAPHIC_SIZELIMIT

This option is used to set the maximum size of the exported graphic in pixels. It may be used to prevent inordinately large graphics from being converted to equally cumbersome output files, thus preventing bandwidth waste.

This setting takes precedence over all other options and settings that affect the size of a converted graphic.

When creating a multi-page TIFF file, this limit is applied on a per page basis. It is not a pixel limit on the entire output file.

Data Type

UInt32

Default

0

15.4.2.18 GraphicWidthLimit

OIT Option ID: SCCOPT_GRAPHIC_WIDTHLIMIT

This option allows a hard limit to be set for how wide in pixels an exported graphic may be. Any images wider than this limit will be resized to match the limit. It should be noted that regardless whether the GraphicHeightLimit option is set or not, any resized images will preserve their original aspect ratio.

Note that this option differs from the behavior of setting the width of graphics by using GraphicWidth in that it sets an upper limit on the image width. Images larger than this limit will be reduced to the limit value. However, images smaller than this width will not be enlarged when using this option. Setting the width using GraphicWidth causes all output images to be reduced or enlarged to be of the specified width.

Data Type
UInt32

Default
0

15.4.2.19 IECondCommentMode
OIT Option ID: SCCOPT_HTML_COND_COMMENT_MODE

Some HTML input files may include "conditional comments", which are HTML comments that mark areas of HTML to be interpreted in specific versions of Internet Explorer, while being ignored by other browsers. This option allows you to control how the content contained within conditional comments will be interpreted by Outside In's HTML parsing code.

Data Type
IECondCommentFlagValues

IECondCommentFlagValues Enumeration
IECondCommentFlagValues can be one or more of the following enumerations ORed together:

- None: Don't output any conditional comment
- IE5: Include the IE5 comments
- IE6: Include the IE6 comments
- IE7: Include the IE7 comments
- IE8: Include the IE8 comments
- IE9: Include the IE9 comments
- ALL: Include all conditional comments

Default
IECondCommentFlagValues.None

15.4.2.20 IgnorePassword
OIT Option ID: SCCOPT_IGNORE_PASSWORD

This option can disable the password verification of files where the contents can be processed without validation of the password. If this option is not set, the filter will return an error.

Currently, only the PST and MDB Filters support this option

Data Type
bool

Default
false

15.4.2.21 ISODateTimes
OIT Option ID: SCCOPT_FORMATFLAGS

When this flag is set, all Date and Time values are converted to the ISO 8601 standard. This conversion can only be performed using dates that are stored as numeric data within the original file.

Data
bool

Default
false

15.4.2.22 LotusNotesDirectory

OIT Option ID: SCCOPT_LOTUSNOTUSDIRECTORY

This option allows the developer to specify the location of a Lotus Notes or Domino installation for use by the NSF filter. A valid Lotus installation directory must contain the file nnotes.dll.

Either Lotus Notes version 8 or Lotus Domino version 8 must be installed on the same machine as OIT. A 32-bit version of the Lotus software must be used if you are using a 32-bit version of OIT. A 64-bit version of the Lotus software must be used if you are using a 64-bit version of OIT.

Data Type
String

Default
None

15.4.2.23 OutputRawtext

An option controlling whether raw text output is generated on the server side. This option must be enabled in order to enable in-browser searching via the Outside In API. This does not affect the search capabilities of the browser itself. Raw text generation is enabled by default.

Data Type
bool

Default
true

15.4.2.24 PageRange

This option indicates whether the whole file or a selected range of pages should be rendered.

Data Type
PageRange

Data
The page range to be exported.

Default
All pages are printed

15.4.2.25 PDFReorderBiDi

OIT Option ID: SCCOPT_PDF_FILTER_REORDER_BIDI

This option controls whether or not the PDF filter will attempt to reorder bidirectional text runs so that the output is in standard logical order as used by the Unicode 2.0 and later specification. This additional processing will result in slower filter performance according to the amount of bidirectional data in the file.

Data Type

PDFReorderBiDiValue

PDFReorderBiDiValue Enumeration

This enumeration defines the type of Bidirection text reordering the PDF filter should perform.

- StandardBiDi: Do not attempt to reorder bidirectional text runs.
- ReorderedBiDi: Attempt to reorder bidirectional text runs.

Default

PDFReorderBiDiValue.StandardBiDi

15.4.2.26 PerformExtendedFI

OIT Option ID: SCCOPT_FIFLAGS

This option affects how an input file's internal format (application type) is identified when the file is first opened by the Outside In technology. When the extended test flag is in effect, and an input file is identified as being either 7-bit ASCII, EBCDIC, or Unicode, the file's contents will be interpreted as such by the export process.

The extended test is optional because it requires extra processing and cannot guarantee complete accuracy (which would require the inspection of every single byte in a file to eliminate false positives.)

Data Type

bool

Data

One of the following values:

- false: When this is set, standard file identification behavior occurs.
- true: If set, the File Identification code will run an extended test on all files that are not identified.

Default

true

15.4.2.27 PostLibraryScripts

List of strings specifying the relative URL of the customer's initialization script. Scripts specified via this option will load after the Outside In libraries, and may access the Outside In Javascript API. The scripts will be referenced in the same order as that of list used to set this option.

Example:

```
<script src="oit.js">
<script src="/config/oitinit.js">
```

Data Type

List<string>

Default

None

15.4.2.28 PreLibraryScripts

List of strings specifying the URL of a script file to be referenced via a <script> tag in the output file. Scripts specified via this option will load before the Outside In libraries. The scripts will be referenced in the same order as that of list used to set this option.

Example:

```
<script src="/jQuery/jQuery-min.js">
<script src="/shims/customstuff.js">
<script src="oit.js">
```

Data Type

List<string>

Default

None

15.4.2.29 ShowArchiveFullPath

This option causes the full path of a node to be returned in "GetRecordInfo" and "GetObjectInfo".

Data Type

bool

Default

false

15.4.2.30 StrokeOutText

This option is used to stroke out (display as graphical primitives) text in an AutoCAD file. Setting this option to false would improve performance, but the visual fidelity may be compromised.

- If the export for the conversion is text only, text is never stroked out.
- If the export is not text only, and the drawing is perspective, text will always be stroked out (regardless of this option). This is due to the fact that in non-text only situations visual fidelity is of importance, and handling of textual objects in perspective drawings is more accurate with stroked out text. If the conversion is non-text only and the drawing is not perspective, this option determines if text should be stroked.

Note that when this option is true, some special characters appear as asterisks or question marks due to limited support of characters for stroking out text.

Data Type
bool

Default
true

15.4.2.31 TimeZoneOffset

OIT Option ID: SCCOPT_TIMEZONE

This option allows the user to define an offset to GMT that will be applied during date formatting, allowing date values to be displayed in a selectable time zone. This option affects the formatting of numbers that have been defined as date values. This option will not affect dates that are stored as text. To query the operating system for the time zone set on the machine, specify TimeZoneOffset_UseNative.

Note: This option does not apply for spreadsheet files.

Daylight savings is not supported. The sent time in msg files when viewed in Outlook can be an hour different from the time sent when an image of the msg file is created.

Data Type
Int32

Default
- 0: GMT time

15.4.2.32 UnmappableCharacter

This option selects the character used when a character cannot be found in the output character set. This option takes the Unicode value for the replacement character. It is left to the user to make sure that the selected replacement character is available in the output character set.

Data Type
UInt16

Default
42

15.4.2.33 URLPathOutput

String option for specifying the base URL for output files generated during the export process. The value of this option is used as a prefix for references to those resources in the primary output file. If the output base URL is "/exports/", an image tag in an exported document would look like this:

```
<img src="/exports/doc01.doc.jpg" />
```

and URLs to the exported files within content.js would be of this form:

```
 {id:"page.1",name:"Page
1",type:"page",oid:"overlay.1",w:"816px",h:"1056px",src:"/exports/output.3.html"}
```

By default, the output file base URL is empty, and all output will be specified without path information.

Data Type
String

Default
None

15.4.2.34 URLPathResources

String option for specifying the base URL for Web View Export scripts, css files and other resources that do not vary with the converted file. The value of this option is used as a prefix for references to those resources in the primary output file. For example, if the resources' base URL is "/oit/", the script reference in the output file would look like this:

```
<script src="/oit/oit.js"></script>
```

By default, this value is empty, meaning all resources must be present in the same directory as the generated output.

Data Type
String

Default
None

15.4.2.35 UseDocumentPageSettings

OIT Option ID: SCCOPT_USEDOCPAGESETTINGS

This option is used to select the document's page layout information when rendering.

If TRUE the document's native (or author selected) page margins, paper size, page scaling and page orientation are used when available from the filter.

The values of the DefaultPrintMargins, RenderGridlines, RenderHeadings, PageDirection, and PageFitMode options are overridden if this option is set to true and the properties associated with those options are specified in the input document. Additionally, print area and page breaks in spreadsheet documents are ignored unless this option is set to true.

If FALSE, the page margins, size, orientation and scaling are set to specific values rather than those in the native document. The page size is forced to 8 1/2" x 11" in portrait orientation, but this may be changed by setting the GraphicHeight and/or GraphicWidth options. The margins are forced 1" all around, but may be changed by setting the defaultMargins option. The scaling for the document will be set to 100%, although this may be changed by setting any of the various scaling options.

It should be noted that this option also affects page orientation for both input spreadsheets and word processing documents.

Data Type
bool

Default
true

15.4.2.36 WebFontPermissions
This option controls the use of "embedded" fonts (see FontReferenceMethod) in the Web View Export output; specifically, whether or not a downloadable (web font) version of the font should be made available to Web View Export output, based on the licensing restrictions indicated within the font itself.

Data Type
WebFontPermissionValues

WebFontPermissionValues Enumeration
WebFontPermissionValues can be one or more of the following enumerations ORed together:

- DefaultPermissions
- RequireInstallable
- RequirePreviewPrint
- RequireEditable
- AllOff: All flags turned off

Default
WebFontPermissionValues.DefaultPermissions

15.4.2.37 WebViewLibraryName
String option specifying the file name of the Web View Javascript library, which will be written into a <script> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.js" will be used. This option allows the Web View Javascript library to be renamed for versioning or other purposes. This option works in combination with the URLPathResources option.

Data Type
String

Default
oit.js

15.4.2.38 WebViewStructure
This option controls the structure of the output files created by Web View Export.

Data Type
WebViewStructureValue

WebViewStructureValue Enumeration
- Flat: HTML content is exported in a single HTML file.
- AjaxChunked: HTML content is spread among multiple output pages.

- AjaxStreamed: HTML content is spread among multiple output pages and the browser may begin viewing the document before the entire output has been produced.

Default

WebViewStructureValue.Flat

15.4.2.39 WebViewStylesheetName

String option specifying the file name of the Web View stylesheet, which will be written into a <link> tag in the generated HTML files for a Web View. If this option is not set, the value "oit.css" will be used. This option allows the Web View stylesheet to be renamed for versioning or other purposes. This option works in combination with the URLPathResources option

Data Type

String

Default

oit.css

15.5 ExportStatus Class

The ExportStatus class provides access to information about a conversion. This information may include information about sub-document failures, areas of a conversion that may not have high fidelity with the original document. When applicable the number of pages in the output is also provided.

- PageCount (Int32) - A count of all of the output pages produced during an export operation.
- StatusFlags (ExportStatusFlags) - Gets the information about possible fidelity issues with the original document.
- SubDocsFailed (Int32) - Number of sub documents that were not converted.
- SubDocsPassed (Int32) - Number of sub documents that were successfully converted.

ExportStatusFlags Enumeration

This enumeration is the set of possible known problems that can occur during an export process.

- NoInformationAvailable: No Information is available.
- MissingMap: A PDF text run was missing the toUnicode table.
- VerticalText: A vertical text run was present.
- TextEffects: A run had unsupported text effects applied. One example is Word Art.
- UnsupportedCompression: A graphic had an unsupported compression.
- UnsupportedColorSpace: A graphic had an unsupported color space.
- Forms: A subdocument had forms.
- RightToLeftTables: A table had right to left columns.
- Equations: A file had equations.

- AliasedFont: The desired font was missing, but a font alias was used.
- MissingFont: The desired font wasn't present on the system.
- SubDocFailed: A subdocument was not converted.

15.6 FileFormat Class

This class defines the identifiers for file formats.

Methods

- GetDescription

  ```
  String GetDescription()
  ```

  This method returns the description of the format.

- GetId

  ```
  int GetId()
  ```

  This method returns the numeric identifier of the format.

- ForId

  ```
  FileFormat ForId(int id)
  ```

  This method returns the FileFormat object for the given identifier.

  id : The numeric identifier for which the corresponding FileFormat object is returned.

15.7 FontAliases Class

FontAliases is a class for providing font matching of unknown fonts.

Constructor

```
FontAliases(Boolean  clearDefaults, Dictionary<string, string> aliasList)
   clearDefaults  Option to clear current list to defaults or empty list
   aliasList      Aliases list as a key-value pair with original name as key
```

Properties

- AliasList (Dictionary<String, String>) - List of font aliases set.
- ClearDefaults (Boolean) - When set to false, the existing alias list is cleared and set it to a list of default aliases. Otherwise the existing alias list is cleared out.

15.8 FontInfo Class

FontInfo is a class to define a font for use in the OutsideIn API.

Constructor

```
FontInfo(String fontface, Int16 height)
   fontface  The name of the font
   height    Size of the font in half points
```

Properties

- Fontface (String) - The name of the font
- Height (Int16) - Size of the font in half points

15.9 FontList Class

FontList is a class for inclusion or exclusion of fonts in exported documents.

Constructor

```
FontList(Boolean  IsExclusion, String[] fonts)
   IsExclusion  If set then accompanying list is an exclusion list
   fonts        List of fonts to include or exclude
```

Properties

- IsExclusion (Boolean) - If set, then accompanying list is an exclusion list.
- FontsList (String[]) - List of fonts to include or exclude.

15.10 HighlightAreaAnnotation Class

The HighlightAreaAnnotation class defines the characteristics of a highlighted area annotation. This class derives from the Annotation class.

Constructors

```
HighlightAreaAnnotation(Int64 userId,  Int64 sectionIx,  Int64 Top,
   Int64 Left,  Int64 Width,  Int64 Height,  Annotation.UnitTypeValue units,
   Single opacity, ColorInfo fillColor,
   HighlightAreaAnnotation.AnnoBorder border,
   HighlightAreaAnnotation.BorderStyleValue borderStyle):
```

Initializes a new instance of the HighlightAreaAnnotation class.

Methods

```
SetBorder(HighlightAreaAnnotation.BorderSidesValue side,
HighlightAreaAnnotation.AnnoBorder)
```

- Parameters:

  side: The side to which this border shall be applied. HighlightAreaAnnotation.BorderSidesValue is an enumeration of the sides of the annotation.

  border: The type of border to be used.

Properties

- **BorderStyle**: The style of the border. (Note: in the initial release of Web View Export, only solid borders are supported)
- **FillColor**: Color to be used to fill the area
- **BottomBorder**
- **LeftBorder**
- **RightBorder**
- **TopBorder**

15.10.1 AnnoBorder Class

The AnnoBorder class defines the characteristics of the borders of annotations.

Constructors

```
AnnoBorder(ColorInfo color, System.Int64thickness)
   colorType: Color of the annotation border
   thicknessType: Thickness of the annotation border
```

Initializes a new instance of the AnnoBorder class.

15.11 HighlightTextAnnotation Class

The HighlightTextAnnotation class applies to characteristics of a highlighted text annotation. This class derives from the Annotation class.

> **Note:** the application of character attributes through highlighting is not supported in the initial release of Web View Export.

Constructors

```
HighlightTextAnnotation(Int64 StartCharCount,
      Int64 EndCharCount,
      CharAttributeValues CharAttrs,
      CharAttributeValues CharMask)
HighlightTextAnnotation(Int64 StartCharCount,
      Int64 EndCharCount,
      ColorInfo Foreground,
      ColorInfo Background)
HighlightTextAnnotation(Int64 StartCharCount,
      Int64 EndCharCount,
      ColorInfo Foreground,
      ColorInfo Background,
      CharAttributeValues CharAttrs,
      CharAttributeValues CharMask)
```

Initializes a new instance of the HighlightTextAnnotation class.

Parameters

- **StartCharCount**: The character count of the starting position
- **EndCharCount**: The character count of the end position
- **Foreground**: The text color of the highlight
- **Background**: The background color of the highlight
- **CharAttrs**: The character attributes to use
- **CharMask**: Character attributes to change

CharAttributeValues Enumeration

This enumeration is the list of all character attributes to apply for the text highlight.

- Normal: Normal text - All attributes off
- Underline: Underline attribute
- Italic: Italic attribute
- Bold: Bold attribute
- StrikeOut: Strike out text
- SmallCaps: Small caps

- Outline: Outline Text
- Shadow: Shadow text
- Caps: All Caps
- Subscript: Subscript text
- Superscript: Superscript text
- DoubleUnderline: Double Underline
- WordUnderline: Word Underline
- DottedUnderline: Dotted Underline
- DashedUnderline: Dashed Underline
- All: All attributes

15.12 Margins Class

The Margins Class is used to describe the page margins.

Constructor

```
Margins(Int64 top,
       Int64 bottom,
       Int64 left,
       Int64 right)
top    Margin from the top edge of the page (in twips)
bottom Margin from the bottom edge of the page (in twips)
left   Margin from the left edge of the page (in twips)
right  Margin from the right edge of the page (in twips)
```

Properties

- Top (Int64) Margin from the top edge of the page (in twips)
- Bottom (Int64) Margin from the bottom edge of the page (in twips)
- Left (Int64) Margin from the left edge of the page (in twips)
- Right (Int64) Margin from the right edge of the page (in twips)

15.13 OutsideIn Class

This is a utility class that creates an instance of an Exporter object on request.

Methods

```
static Exporter NewLocalExporter()
```

This method creates an instance of an Exporter object. It returns a newly created Exporter object.

Properties

InstallLocation (DirectoryInfo): Location of the Outside In binaries. This value is required if the Outside In binaries reside in a directory other than the application directory. Also, this can only be set once and should be set before the first Exporter object is created.

15.14 OutsideInException Class

This is the exception that is thrown when an Outside In Technology error occurs.

This class derives from the Exception class. This class has no public methods or properties except those of the parent Exception class.

15.15 PageInfo Class

PageInfo is a class for defining page dimensions.

Constructor

```
PageInfo(PageInfo.PageSizeUnitsValues units,
             Single width,
             Single height)
units      Units used to specify width and height
width    Width of the page
height   Height of the page
```

Properties

- Units: Units used to specify width and height
- Width: Width of the page
- Height: Height of the page

PageInfo.PageSizeUnitsValues Enumeration

PageSizeUnitsValues is an enumeration of the various units that can be used to specify the width and height of a page.

- Inches: Units are in Inches
- Points: Units are in Points (1/72th of an inch)
- Centimeters: Units are in Centimeters
- Picas: Units are in Picas (1/6th of an inch)

15.16 PageRange Class

PageRange is a class for defining page ranges for exporting purposes.

Constructors

```
PageRange()
```

Creates an instance of the PageRangeObject for printing all pages.

```
PageRange(Int32 StartPage)
StartPage  Starting page number of the print range
```

Creates an instance of the PageRangeObject for printing from a page until end of document.

```
PageRange(
           Int32  StartPage,
           Int32  StopPage)
StartPage  Starting page number of the print range
StopPage   End page number of the print range
```

Creates an instance of the PageRangeObject for printing a range of pages.

Properties

- PrintAll (Boolean) - If set to true, all pages of the document will be printed
- StartPage (Int32) - The start page of the print range. 0 indicates printing will begin with the first page of the document.
- StopPage (Int32) - The last page of the print range. 0 indicates the last page at the end of the document.

15.17 StampAnnotation Class

StampAnnotation is the class for defining stamp annotations. This class derives from the Annotation class.

The StampAnnotation type exposes the following members.

Constructors

```
StampAnnotation(long userId,
   Int64 sectionIx,
   Int64 Top,
   Int64 Left,
   Int64 Width,
   Int64 Height,
   Annotation.UnitTypeValue units,
   Single opacity,
   StampAnnotation.SizingMethodValue sizemode,
   String stampName)
userId     User Data
sectionIx  0-based page/sheet/image/slide index
Top        Top coordinate or row
Left       Leftmost coordinate or column
Width      Width of area in coordinates or columns
Height     Height of area in coordinates or rows
Units      Unit type
Opacity    Opacity of the annotation. Range 0.0 - 1.0; 0==invisible
Sizemode   Image sizing method
stampName  Name of the stamp
```

Properties

SizeMode: Sizing method of the source image

StampName: Registered name of stamp image

SizingMethodValue Enumeration

The SizingMethodValue is an enumeration of the sizing methods that can be applied to the image.

15.18 TreeRecord Class

TreeRecord provides information about an archive record. This is a read-only class where the technology fills in all the values.

Properties

- bIsDirectory (Boolean) A value of true indicates that the record is a directory node.

- dwFileSize (Int32) File size of the tree record
- dwTime (Int32) Time the tree record was created
- nNodeNum (Int32) Serial number of the tree record in the archive
- strNodeName (String) The name of the tree record