Oracle® Outside In XML Export

Developer's Guide

Release 8.5.4
E80947-01
May 2018

Oracle Outside In XML Export Developer's Guide, Release 8.5.4

E80947-01

Primary Author: Nirmala Suryaprakasha

Contents

Preface

This document describes the installation and usage of the Outside In XML Export Software Developer's Kit (SDK).

Audience

This document is intended for developers who are integrating Outside In XML Export into Original Equipment Manufacturer (OEM) applications.

Related Documents

The complete Oracle Outside In Technology documentation set is available from the Oracle Help Center at http://www.oracle.com/pls/topic/lookup?ctx=oitlatest&id=homepage.

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.
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.

Part I
Getting Started with XML Export

This section provides an introduction to the SDK.

Part I contains the following chapters:

1
Introduction

This chapter provides an introduction to XML Export. XML Export allows developers to implement sophisticated text extraction from standard business documents.

Note:

For new functionality information, see What's New guide.

With the current version of XML Export, an application can access documents through a C, Java, or .NET API. Included with XML Export is the powerful Flexiondoc schema.

There may be references to other Outside In Technology SDKs within this manual. To obtain complete documentation for any other Outside In product, see Middleware documentation page and click Outside In Technology link below.

This chapter includes the following sections:

- What Does This Technology Do?
- Architectural Overview
- Definition of Terms
- Directory Structure
- How to Use XML Export

1.1 What Does This Technology Do?

XML Export can normalize all of a document's content to the Flexiondoc schema, provided in the form of a DTD and an XML schema.

Note:

All XML Export output formats are UTF-8 encoded Unicode text.

1.1.1 Flexiondoc Schema

The Flexiondoc schema is designed to provide extremely dense, rich XML versions of input documents, enabling powerful applications such as document assembly, portals and content management systems.

Here are some of the schema's primary features:

- Translation of documents to XML, with all characters translated to Unicode
- A common interface to more than 600 file formats

- Access to document properties
- Support for word processor, spreadsheet, graphic, and archive formats
- Support for embeddings
- Special tags are created for hyperlinks, bookmarks, and sub-documents

1.2 Architectural Overview

The basic architecture of Outside In technologies is the same across all supported platforms:

Filter/Module	Description
Input Filter	The input filters form the base of the architecture. Each one reads a specific file format or set of related formats and sends the data to OIT through a standard set of function calls. There are more than 150 of these filters that read more than 600 distinct file formats. Filters are loaded on demand by the data access module.
Export Filter	Architecturally similar to input filters, export filters know how to write out a specific format based on information coming from the chunker module. The export filters generate XML.
Export	The Export module implements the export API and understands how to load and run individual export filters.
Data Access	The Data Access module implements a generic API for access to files. It understands how to identify and load the correct filter for all the supported file formats. The module delivers to the developer a generic handle to the requested file, which can then be used to run more specialized processes, such as the Export process.
Schema	Schemas provide a means for defining the structure, content and semantics of XML documents. XML Export ships with the Flexiondoc schema. Schemas can be presented in the form of a DTD (Document Type Definition) or XML Schema (schema). The Flexiondoc schema is provided in both forms.

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 file being written: FlexionDoc, XML, GIF, JPEG, and PNG.
Data Access Module	The core of Outside In Data Access, in the *SCCDA* library.
Data Access Submodule (also referred to as "Submodule")	This refers to any of the 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 Data Access Common Functions). Each Document Handle may have any number of Subhandles.

Term	Definition
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 Outside In product has an sdk directory, under which there is a subdirectory for each platform on which the product ships (for example, xx/sdk/xx_win-x86-32_sdk). Under each of these directories are the following two subdirectories:

- **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, like freetype.
- **sdk** - Contains the other subdirectories that used to be at the root-level of an sdk (common, lib (windows only), resource, samplefiles, and samplecode (previously samples). In addition, one new subdirectory has been added, demo, that holds all of the compiled sample apps and other files that are needed to demo the products. These are files that the customer should not redistribute (.cfg files, exportmaps, etc.).

In the root platform directory (for example, xx/sdk/xx_win-x86-32_sdk), 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:

- **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.

1.5 How to Use XML Export

Here's a step-by-step overview of how to export a source file to XML.

1. Call DAIniExt to initialize the Data Access technology. This function needs to be called only once per application. If using threading, then pass in the correct ThreadOption.
2. Set any options that require a NULL handle type (optional). Certain options need to be set before the desired source file is opened. These options are identified by requiring a NULL handle type. They include, but aren't limited to:
   - SCCOPT_FALLBACKFORMAT
   - SCCOPT_FIFLAGS
   - SCCOPT_TEMPDIR
3. Open the Source File. DAOpenDocument is called to create a document handle that uniquely identifies the source file. This handle may be used in subsequent calls to the EXOpenExport function or the open function of any other Data Access Submodule, and will be used to close the file when access is complete. This allows the file to be accessed from multiple Data Access Submodules without reopening.
4. Set the Options. If you require option values other than the default settings, call DASetOption to set options. Note that options listed in the Options Guide as having "Handle Types" that accept VTHEXPORT may be set any time before EXRunExport is called. See "DASetOption" for more information on options and how to set them.
5. Open a Handle to XML Export. Using the document handle, EXOpenExport is called to obtain an export handle that identifies the file to the specific export product. This handle will be used in all subsequent calls to the specific export functions. The dwOutputId parameter of this function is used to specify that the output file type should be set to FI_XML_FLEXIONDOC_LATEST.
6. Export the File. EXRunExport is called to generate the output file(s) from the source file.
7. Close the Handle to XML Export. EXCloseExport is called to terminate the export process for the file. After this function is called, the export handle will no longer be valid, but the document handle may still be used.
8. Close the Source File. DACloseDocument is called to close the source file. After calling this function, the document handle will no longer be valid.
9. Close XML Export. DADeInit is called to de-initialize the Data Access technology.

2 Implementation Issues

This chapter describes the implementation issues specific to using XML Export. The following issues are addressed:

- Running in 24x7 Environments
- Running in Multiple Threads or Processes

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 Introduction.

3
Sample Applications

This chapter describes the sample applications for XML Export. 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. This chapter includes the following sections:

- Building the Samples on a Windows System
- An Overview of the Sample Applications
- Accessing the SDK via a Java Wrapper

3.1 Building the Samples on a Windows System

Microsoft Visual Studio 2010 files are provided for building each of the sample applications.

Because .vcxproj files may not pick up the right compiler on their own, you need to make sure that you are building with the correct configuration in Visual Studio 2010 or higher.

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

See UNIX Implementation Details, for specific information about building the sample applications on your UNIX OS.

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:

3.2.1 *sample

The name of this sample application varies according to product (**xxsample** for XML Export).

The following is a basic implementation that uses the default settings for every option.

```
xxsample Inputfile Outputfile
```

This sample app provides a very simple demonstration of creating FlexionDoc output.

3.2.2 export (Windows Only)

This application was designed to facilitate the testing of the software and should not be assumed to be of commercial quality.

> **Note:**
>
> No default options are set at initial runtime. The time the software is used, click the **Options** button and set the options. Failure to do this generates export errors.

The application allows the user to run a single source file. The user can choose the source file, an output file and set the various options. If you are going to rebuild this application, be sure to set your INCLUDE and LIB paths to the \COMMON and \LIB directories respectively.

3.2.2.1 The export Main Window

The Main Window contains the following fields.

- Output Format menu: This menu allows the user to select the type of output to generate. An entry for the format(s) you license will appear in this drop-down menu
- **Options** button: This opens up a new dialog with one or more tabs exposing the options for the selected product.
- Source document field: This is the document to be exported. Use the Browse button to pick the source file, or type in the path name.
- 'Export to' Field: This is the initial resulting output file. Type in a file name or use the Browse button to choose a file. Other output files are named based on the one chosen here.
- **Delete** button: Clicking this button deletes all files generated by the last export, listed in the Status: field. This is useful when multiple output files are produced because the default naming rules do not overwrite an existing file. If you run Export over and over again with the same output file name, you can produce a large number of files. Pressing **Delete** before each export solves this problem.
- 'After Export, view output file with default application' Check Box: If the export was successful, checking this box launches the initial output file in the application associated with the output flavor's default extension.
- **Export** button: Click to start the export process once you've determined the export settings.
- **Exit** button: Click to close the Export application.

3.2.3 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_XML_FLEXIONDOC_LATEST before running the software.

3.2.4 extract_archive

extract_archive demonstrates using the DATree API to extract all nodes in an archive.

The application is executed from the command line and takes two parameters, the name of the input file and the name of an output directory for the extracted files:

```
extract_archive input_file output_directory
```

3.2.5 xxredir (XML Export)

This sample application is based on the **exsimple** sample application. It is designed to demonstrate how to use redirected IO and callbacks when using the software. It takes the same arguments and command line structure as exsimple and the same configuration files can be used. See "exsimple" for details.

3.3 Accessing the SDK via a Java Wrapper

The ExJava Java wrapper, working in tandem with the exporter sample application, provides a working example of one method of interfacing with Oracle's C-based SDK products from a Java application. Export.jar is a Java API wrapper used by a Java application to control the exporter executable and set conversion options. exporter is a C-based executable which performs conversions using the modules in the Outside In SDK.

The exporter executable should be placed in the root directory of the Outside In SDK being used. If more than one Outside In SDK is being used, the contents of each SDK should be unpacked to the same root directory. Export.jar should be placed somewhere in your classpath.

On UNIX systems this sample application must be run from the directory containing the Outside In technology.

Java version 1.6 or higher is required to run this sample application.

This section covers the following topics:

3.3.1 The ExJava Wrapper API

The JavaDocs documentation for the Java API is provided in the /sdk/samplecode/ExJava/docs directory. Conversion options are set using the ExportProperties.

Additionally, the appropriate .cfg file for the ExportTest sample application found in the Examples/ExportTest directory may provide further insight as to what properties are available and how they correspond to options and values for options.

The Export.jar and its source code can be found in the Java API directory. Place Export.jar somewhere in your classpath. In order to use the ExportTest sample application (which demonstrates how a Java application can use the ExJava API) without modifying your system configuration or the ExJava sample application, you should place the Export.jar file in the root directory of the Outside In SDK product you are using.

3.3.2 The C-Based Exporter Application

This is a standalone executable that runs out of process from the Java API. The Java API controls the conversion through command line parameters that are passed to the executable. After the conversion completes, the executable returns a conversion status code to the Java API. The command line parameters are base-64 encoded to allow for the use of Unicode encoded paths.

As the exporter executable is a C-based application, you will need to make sure the Java API can find the version of exporter appropriate for the platform you are using. Generally, and specifically for the purpose of using the ExportTest sample application, the correct executable should be copied to the root directory of the Oracle export SDK product you are using.

A compiled version of the C exporter program is included in the SDK with the rest of the Outside In binaries. The source for exporter is located in the /sdk/samplecode/ExJava/exporter directory.

The current implementation of ExJava may not produce an error if it cannot find the exporter application. This known issue may be corrected in a future version of ExJava.

3.3.3 Compiling the Executables

A Microsoft Visual Studio 6.0 project file and a UNIX makefile are provided in /sdk/samplecode/ExJava/exporter/win and /sdk/samplecode/ExJava/exporter/unix, respectively, so that you can modify the exporter executable or compile it for a platform other than those for which compiled versions of exporter are provided. If you unpacked the ExJava package into the root directory of one of Oracle's export SDK products, you should be able to use the Visual Studio Project and makefile as is. Otherwise, you will need to edit them in order to provide paths to the Oracle export SDK include and library files.

If you are compiling ExJava for use on the Solaris platform, make sure your LD_LIBRARY_PATH contains the Outside In SDK path before trying to build the exporter module.

3.3.4 The ExportTest Sample Application

ExportTest is an example of how a Java developer could use the ExJava wrapper to use one of the Outside In SDKs. The following is a list of the components that should be placed in the root directory of the Outside In SDK you are using in order to run this sample application:

1. Export.jar (from the Java API directory)
2. Exporter module for the platform you wish to use (located in the /sdk/samplecode/ExJava/exporter/win or /sdk/samplecode/ExJava/exporter/unix directory, depending on which platform you are using)
3. xx.cfg (also in Examples/ExportTest directory)
4. If you are running ExportTest on a UNIX system, make sure to edit the .cfg file so it reflects the correct name of the exporter module you renamed.
5. ExportTest.jar (also in Examples/ExportTest directory)
6. The appropriate batch file to run the ExportTest application (ExportTest.bat for Windows and ExportTest.sh for UNIX, both located in the Examples/ExportTest directory)

Once these files are properly copied, execute the batch file with the name/path of an input file to convert, the name for the base output file and the name of the configuration file to use for setting conversion options.

ExportTest.jar uses the contents of the configuration file to determine what option/value pairs it should use when doing the conversion. It is not necessary to use a configuration file when developing your own application if you so choose not to.

3.3.5 An Example Conversion Using the ExJava Wrapper

This is a simple outline of the steps for using the ExJava wrapper on a Windows system to convert a Word document called MyWordDoc.Doc. If you are using a UNIX system, see UNIX Implementation Details for information about properly setting up your environment to use the Outside In SDK:

1. Edit the .cfg file and make sure outputid is set to the FI* value appropriate for the Outside In product you've licensed. Alter any other parameters in the .cfg file as needed then save the file.
2. Execute the following command. The sample command below assumes XML as the export type. Change this type accordingly:

```
ExportTest.bat myworddoc.doc output.xml xx.cfg
```

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

This chapter discusses the Windows implementation of the XML Export SDK. The Windows implementation of this software is delivered as a set of DLLs.
For a list of the currently supported platforms, see Outside In Technology and click links under Certified Platforms and Supported Formats from Get Started page.

This chapter includes the following sections:

- Installation
- Libraries and Structure
- The Basics
- Changing Resources

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 are versions of this package for the x86 and x64 versions of Windows. This can be downloaded from www.microsoft.com/downloads, by searching on the site for the following packages:

- vcredist_x86.exe
- vcredist_x64.exe

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

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 Windows version of the technology.
\sdk\common	Contains the C include files needed to build or rebuild the technology.
\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 input files authored in a variety of popular graphics, word processor, compression, spreadsheet and presentation applications, designed to exercise XML Export.
\sdk\XXSchemaDocs	Contains the current Flexiondoc manual.

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 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 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	ImageExport	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	

Library	Description	HTML Export	ImageExport	PDF Export	Search Export	XML Export	Web View Export
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				

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
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.dll	Import filters for premier graphics formats	X	X	X	X	X	X
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

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
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

This software is based on the Outside In Viewer Technology (or simply "Viewer Technology"). 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:

- *.f = Filter lists
- *.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 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

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 ANSI1252 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 Changing Resources

Outside In XML 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 environment.

Note:

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

5
UNIX Implementation Details

This chapter discusses the UNIX implementation of XML Export. 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 Outside In Technology and click links under Certified Platforms and Supported Formats from Get Started page.

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/common	Contains the C include files needed to build or rebuild the technology.
/sdk/demo	Contains the compiled executables of the sample applications.
/sdk/resource	Contains localization resource files. See Changing Resources for details.
/sdk/samplecode	Contains a subdirectory holding the source code for a sample application. See Sample Applications for more details.

Directory	Description
/sdk/samplefiles	Contains sample input files authored in a variety of popular graphics, word processor, compression, spreadsheet and presentation applications, designed to exercise XML Export.
/sdk/XXSchemaDocs	Contains a Flexiondoc manual, version 5.4

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 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. See Environment Variables for more details.

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. This library is only supported on Linux (32- and 64-bit Intel), Solaris (32-bit SPARC), HP-UX (32-bit RISC), and AIX (32-bit PPC).	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

Library	Description	HTML Export	Image Export	PDF Export	Search Export	XML Export	Web View Export
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
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. This library is only supported on Linux (32- and 64-bit Intel), Solaris (32-bit SPARC), HP-UX (32-bit RISC), and AIX (32-bit PPC).	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	
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		
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. See Sample Applications for a complete description of the 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 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 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:

- *.f: Filter lists
- *.d: Display engine list
- *.opt: Persistent options

The technology does not actually use the list of default options created by the Viewer Technology.

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 Viewer Technology and/or other Viewer Technology-based products are installed on a single system. The filenames are built from an 11-character string derived from the directory the 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 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 /sdk/common directory of the technology.

5.5 Runtime Considerations

The following is information to consider during run-time.

5.5.1 X Server Requirement

Note:

The X Server requirement can be eliminated by setting the SCCOPT_RENDERING_PREFER_OIT option to TRUE.

Access to a running X Windows server and the presence of Motif (or LessTif on Linux) are required to convert from vector formats on UNIX systems. Examples of vector graphics files include CAD drawings and presentation files such as Power Point 97 files. Bitmap graphic conversion (handled in XML Export by the libde_bmp.* engine) does not require access to a running X Windows server. Examples of bitmap file formats include GIF, JPEG, TIFF, and Windows BMP files.

A runtime check for the presence of X libraries is performed to accommodate system with and without available X servers. This check looks on the system-specific library path variable for the X libraries. If the X libraries are not found, this product does not perform vector graphics conversion.

Be sure to set the $DISPLAY environment variable before running this product when non-raster/vector graphic conversion is needed. This is especially important to remember in situations such as CGI programs that start with a limited environment.

For example, when running the technology from a remote session, setting DISPLAY=:0.0 tells the system to use the X Windows server on the console.

5.5.2 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, 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.

5.5.3 Machine-Dependent Graphics Context

The system uses a machine configuration dependent graphics context to render some images. The number of colors available in the systems graphics context is a particularly important limiting factor. For example, if the video driver for a system running Outside In is set up to display 256 colors, images produced on that system would be limited to 256 colors.

- For all vector image formats that HX converts, we require that the X11 display support either 1 bit, 4 bits, 8 bits, 24 bits, or 32 bits.
- If SCCOPT_RENDERING_PREFER_OIT = TRUE on UNIX then we're using internal rendering of vector formats, and we don't use the X11 display.
- Raster image formats when converted do not need the X11 display, so are not sensitive to the bit depth of the display.

Note:

SCCOPT_RENDERING_PREFER_OIT is only supported on Linux x86-32 and Solaris Sparc-32 platforms.

5.5.4 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`

5.5.5 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 use 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 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 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.
$GDFONTPATH	This variable must be set. It includes one or more paths to fonts for use with Outside In's internal graphics rendering code.

Variable	Description
$HOME	Must be set to allow the system to write the option, filter and display engine lists. See "Information Storage" for details.

5.7 Changing Resources

All of the strings used in the UNIX versions of 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 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 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 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.

5.8 HP-UX Compiling and Linking

The libsc_ex.sl and libsc_da.sl libraries are the only ones that must be linked with your application. 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, shl_load).

The shared libraries are dependent on the presence of the X libraries Xm, Xt and X11 if vector graphics support is required. It is the application developer's responsibility to ensure that the needed functions from these libraries are present before the product libraries are used.

The following are example command lines used to compile the sample application **exsimple** from the /sdk/samplecode/unix directory. The command lines are separated into sections for HP-UX and HP-UX on Itanium (which requires GCC). 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 and any required X 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 that the compiler and linker can locate all required files.

5.8.1 HP-UX on RISC

```
cc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c +DAportable -Ae -
I/usr/include -I../../common -L../../demo -L/usr/lib -lsc_ex -lsc_da -Wl,+s,
+b,'$ORIGIN'
```

5.8.2 HP-UX on Itanium (64 bit)

```
cc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c +DD64 -I../../common -
L../../demo -lsc_ex -lsc_da -DUNIX_64 -Wl,+s,+b,'$ORIGIN'
```

5.9 IBM AIX Compiling and Linking

All libraries should be installed into a single directory and the directory must be included in the system's shared library path ($LIBPATH). $LIBPATH *must* be set and must point to the directory containing the Outside In Technology.

Outside In Technology has been updated to increase performance, at a cost of using more memory. It is possible that this increased memory usage may cause a problem on AIX systems, which can be very conservative in the amount of memory they grant to processes. If your application experiences problems due to memory limitations with Outside In, you may be able to fix this problem by using the "large page" memory model.

If you anticipate viewing or converting very large files with Outside In technology, we recommend linking your applications with the -bmaxdata flag. For example:

```
cc -o foo foo.c -bmaxdata:0x80000000
```

If you are currently seeing "illegal instruction" errors followed by immediate program exit, this is likely due to not using the large data model.

The shared libraries are dependent on the presence of the X libraries Xm, Xt and X11 if vector graphics support is required. It is the application developer's responsibility to ensure that the needed functions from these libraries are present before the product libraries are used.

The following is an example command line used to compile the sample application exsimple from the /sdk/samplecode/unix 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 and any required X 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 that the compiler and linker can locate all required files. Developers need to pass -brtl to the linker to list libraries in the link command as dependencies of their applications.

Developers may need to use the -qcpluscmt flag to allow C++ style comments.

5.9.1 IBM AIX (32-bit pSeries)

```
gcc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c -I../../common -
L../../demo -lsc_ex -lsc_da -DFUNCPROTO -Wl, -brtl
```

5.10 Linux Compiling and Linking

This section discusses issues involving Linux compiling and linking.

5.10.1 Library Compatibility

This section discusses Linux compatibility issues when using libraries.

5.10.1.1 Motif Libraries

Problems can be seen when using Export products and trying to convert graphics files. For example, zero-byte graphics files are generated if the technology cannot find the proper Motif library. You can check to see if this is the case by running the following command:

```
ldd libos_xwin.so
```

This prints a list of the dependencies that this library has. If the line for the Motif library is similar to the following then your system may not have a compatible Motif library:

```
libXm.so.3 => not found
```

The solution is to install a compatible Motif library and use it to build your application. Often, the installation discs for your particular Linux platform have the proper libraries. If your installation discs do not have the libraries, instructions for downloading a binary rpm can be found at `http://rpmfind.net/linux/RPM`.

If you are doing development, you must use the proper header files, as well.

The following is a list of the Motif library versions used by Oracle when building and testing the Outside In binaries.

- x86 Linux - OpenMotif v. 2.2.3
- zSeries Linux - OpenMotif v. 2.2.3
- Itanium Linux - OpenMotif v. 2.1.30

If a directory needs to be specified for the compiler to find the shared libraries, the $LD_LIBRARY_PATH environment variable is recommended. This prevents the compiler from hard-coding the library's current directory into the executable as the only directory to search for the library at run time. Instead, the system first searches the directories specified by $LD_LIBRARY_PATH for the library.

5.10.1.2 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 Outside In operation.

Distribution	Compiler Version	GLIBC Version
x86 Linux	3.3.2	libc.so.6 (2.3.2 or newer)
Itanium Linux	3.3.2	libc.so.6 (2.3.2 or newer)
zSeries Linux	3.3.6	libc.so.6 (2.3.2 or newer)

5.10.1.3 Other Libraries

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

- libXm.so.3 (in particular, libXm.so.3.0.2 or newer, due to issues in OpenMotif 2.2.2)
- libXt.so.6
- libstdc++.so.6
- libgcc_so.1

libgcc_s.so.1 was introduced with GCC 3.0, so any distribution based on a pre-GCC 3.0 compiler does not include libgcc_s.so.1.

5.10.2 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 shared libraries are dependent on the presence of the X libraries Xm, Xt and X11 if vector graphics support is required. It is the application developer's responsibility to ensure that the needed functions from these libraries are present before the product libraries are used.

The following are example command lines used to compile the sample application **exsimple** from the /sdk/samplecode/unix 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 and any required X 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.

The -L/usr/X11R6/lib option is also available.

5.10.2.1 Linux 32-bit, including Linux PPC

```
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 -Wl,-rpath,../../demo -
Wl,-rpath,'${ORIGIN}'
```

5.10.2.2 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}'
```

5.10.2.3 Linux zSeries

```
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 -Wl,-rpath,../../demo -
Wl,-rpath,'${ORIGIN}'
```

5.11 Oracle Solaris Compiling and Linking

> **Note:**
>
> These products do not support the "Solaris BSD" mode.

All libraries should be installed into a single directory. The libsc_ex.so, and libsc_da.so libraries must be linked with your application. It 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 shared libraries are dependent on the presence of the X libraries Xm, Xt and X11 if vector graphics support is required. It is the application developer's responsibility to ensure that the needed functions from these libraries are present before the product libraries are used.

The following examples command line used to compile the sample application **exsimple** from the /sdk/samplecode/unix 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 and any required X 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 that the compiler and linker can locate all required files.

Developers may need to use the -xcc flag to allow C++ style comments.

5.11.1 Oracle Solaris SPARC

```
cc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c -I/usr/include -
I/usr/dt/share/include -I../../common -L../../demo -L/usr/lib -L/lib -lsc_ex -lsc_da
-Wl,-R,../../demo -Wl,-R,'${ORIGIN}'
```

Note: When running the 32-bit SPARC binaries on Solaris 9 systems, you may see the following error:

```
ld.so.1: simple: fatal: libm.so.1: version `SUNW_1.1.1' not found
(required by file ./libsc_vw.so)
```

This is due to a missing system patch. Please apply one of the following patches (or its successor) to your system to correct.

- For Solaris 9 - Patch 111722-04

5.11.2 Oracle Solaris x86

Note:

Your system will require Solaris patch 108436, which contains the C++ library libCstd.so.1.

```
cc -w -o ../exsimple/unix/exsimple ../exsimple/unix/exsimple.c -I/usr/include -
I/usr/dt/share/include -I../../common -L../../demo -L/usr/lib -L/lib -lsc_ex -lsc_da
-Wl,-R,../../demo -Wl,-R,'${ORIGIN}'
```

6

Data Access Common Functions

This chapter describes common data access functions for XML Export. The Data Access module is common to all Outside In technologies. 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 Deprecated Functions

DAInit and DaThreadInit have both been deprecated. DAInitEx now replaces these two functions. All new implementations should use DAInitEX, although the other two functions will continue to be supported.

6.2 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, the 32-bit versions of Linux x86 and Solaris SPARC, Linux x64 and Solaris SPARC 64. 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.

6.3 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..

6.4 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.

  > **Note:**
  >
  > The values used within IOTYPE_ARCHIVEOBJECT, IOTYPE_LINKEDOBJECT, and IOTYPE_OBJECT may change if different options are applied, with different versions of the technology, or after patches are applied.

  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. See Redirected IO for more information.

  – IOTYPE_ARCHIVEOBJECT: All platforms. Opens an embedded archive object for data access. pSpec points to a structure IOSPECARCHIVEOBJECT (see "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 "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.

6.4.1 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;
```

6.4.2 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;
```

6.5 DAOpenSubdocumentById

Allows an embedding to be opened using the integer value of the object_id attribute from the locator element.

Prototype

```
DAERR DAOpenSubdocumentById(
   VTHDOC     hDoc,
   VTLPHDOC   lphDoc,
   VTDWORD    dwSubdocumentId,
   VTDWORD    dwFlags);
```

Parameters

- hDoc: The document handle for the document containing the locator.
- lphDoc: Receives the document handle for the embedding.
- dwSubdocumentId: The integer value of the object_id attribute from the locator.
- dwFlags: Must be set to 0.

6.6 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.

6.7 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.

6.8 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.

6.9 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.

6.10 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, then 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.

6.11 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

6.12 DAGetObjectInfo

This function returns information about the document or object pointed to by hDoc. The object may be an embedded object, a linked object, or a compressed file.

```
DAERR DAGetObjectInfo(
   VTHDOC      hDoc,
   VTDWORD     dwInfoId,
   VTLPVOID    pInfo);
```

Parameters

- hDoc: The handle returned by DAOpenDocument.

- dwInfoId: The identifier of the requested information. Can be any of the following values:
  - DAOBJECT_NAME_A: Retrieves the name of the object, in 8-bit characters. pInfo points to a buffer of size DA_PATHSIZE.
  - DAOBJECT_NAME_W: Retrieves the name of the object in Unicode characters. pInfo points to a buffer of 16 bit characters of size DA_PATHSIZE.
  - DAOBJECT_FORMATID: Retrieves the file ID of the object. pInfo points to a VTDWORD value.
  - DAOBJECT_COMPRESSIONTYPE: Retrieves an identifier of the type of compression used to store the object, if known. pInfo points to a VTDWORD value.
  - DAOBJECT_FLAGS: Retrieves a bitfield of flags indicating additional attributes of the object. pInfo points to a VTDWORD value. Possible flag values include DAOBJECTFLAG_PARTIALFILE (would not normally exist outside the source document), DAOBJECTFLAG_PROTECTEDFILE (encrypted or password protected), DAOBJECTFLAG_LINKTOFILE (indicates that an OLE object is linked to the file and a corresponding file is not found on the host machine), DAOBJECTFLAG_UNIDENTIFIEDFILE (indicates that an object could not be identified), and DAOBJECTFLAG_UNSUPPORTEDCOMP (compressed with an unsupported compression), and DAOBJECTFLAG_ARCKNOWNENCRYPT (see note below).
- pInfo: Destination of the requested information. The possible types are described in the preceding section about dwInfoId.

Note:

DAOBJECTFLAG_ARCKNOWNENCRYPT indicates that the object is protected by a known encryption. It can be accessed after the correct credentials (password and/or Lotus Notes id file) are provided through the File Access Callback. For details, see DASetFileAccessCallback.

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.

6.13 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

6.14 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.his returned.
- DAERR_BADPARAM: The selected file does not contain an archive section, or the requested record does not exist.

6.15 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.

6.15.1 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 to retrieve information about. 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.

> **Note:**
>
> DAOBJECTFLAG_ARCKNOWNENCRYPT indicates that the object is protected by a known encryption. It can be accessed after the correct credentials (password and/or Lotus Notes id file) are provided through the File Access Callback. See DASetFileAccessCallback.

- 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.

6.16 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.

6.17 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: 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.

6.18 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.

6.19 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 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 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 Outside In to abort.

The status callback function has two return values defined:

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

6.20 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 Outside In that the requested information is provided.
- SCCERR_CANCEL: Tells 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);
   VTHANDLE hUnique,
   VTLPVOID pAppData)
```

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. *DASetFileAccessCallback* applies to filters that support password protected files. Check filter for any or all calls to *UTGetFileAccess* in filters and core modules.

Only Microsoft Office binary (97-2003), Microsoft Office 2010-2013, Microsoft Outlook PST 97-2016, Lotus NSF, PDF (with RC4 encryption), and 7zip (with AES 128 & 256 bit, ZipCrypto) are currently supported.

Passwords for PST/OST files must be in the Windows single-byte character set. For example, Cyrillic characters should use the 1252 character set. For PST/OST files, Unicode password characters are not supported.

7

Export Functions

This chapter outlines the basic functions used to initiate the conversion of documents using the XML Export SDK.
This chapter describes the following functions:

- EXOpenExport
- EXCALLBACKPROC
- EXCloseExport
- EXRunExport
- EXExportStatus

7.1 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);
```

phExport is *not* a file handle.

Parameters

- hDoc: A handle that identifies the source file, created by DAOpenDocument. XML Export does this internally (when exporting graphics). 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 should be set to FI_XML_FLEXIONDOC_LATEST.
- 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. A pointer to a BASEIO structure filled in by your application. This must not be set to NULL or conversion fails.
- pSpec: Initial output file location specification. The form of this data depends on the value of the dwSpecType parameter (see above). This is either a pointer to a buffer or NULL.
- 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. See the definition for EXCALLBACKPROC in "EXCALLBACKPROC" for more details. This parameter may be set to NULL if the developer does not wish to handle callbacks.
- dwCallbackData: This parameter ispassed 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.

Return Values

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

7.2 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. See Callbacks for information about supported callbacks.
- pCommandOrInfoData: Data associated with dwCommandOrInfoId. See Callbacks for information about supported 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.

7.3 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.

7.4 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.

7.5 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 
EXSTATUSVERSION12      */
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 */
VTBOOL bTypeThreeFont;               /* a Type 3 Font was encountered */
VTBOOL bUnsupportedShading;        /* an unsupported shading pattern was encountered
*/
VTBOOL bInvalidHTML;                  /* An HTML parse error, as defined by the W3C, 
was encountered. */
VTBOOL bVectorObjectLimit;            /* This flag does not apply to XML Export */
VTBOOL bInvalidAnnotationNotApplied;  /* This flag does not apply to XML Export */
} EXSTATUSINFORMATION;
```

#define EXSTATUSVERSION2 0X0002

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.

bVectorObjectLimit applies only to WebView Export, and bInvalidAnnotationNotApplied applies only to Image Export, PDF Export, and Web View Export.

8

Redirected IO

This chapter covers the use of Redirected IO for XML Export. 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.

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.

Note:

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. See "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 "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 Outside In technology. See "File Types That Cause IOGETINFO_GENSECONDARY" for a list of these file types.

  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 Outside In IO types or totally redirect the IO for the secondary file, as well. For more details, see "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 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 "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

This chapter describes the use of callbacks in XML Export. Callbacks allow the developer to intervene at critical points in the export process. 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. Read more about the callback procedure and the EXOpenExport function call in EXOpenExport.

A second callback function, DASetStartCallback, can provide information about the progress of a file conversion. See Data Access Common Functions for more details.

This chapter includes the following sections:

- EX_CALLBACK_ID_CREATENEWFILE
- EX_CALLBACK_ID_GRAPHICEXPORTFAILURE
- EX_CALLBACK_ID_NEWFILEINFO

9.1 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.

These include all output text and graphics files that are created. However, it does not include the already open initial file passed to EXOpenExport, 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_XML_FLEXIONDOC_LATEST.
- 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.
  - CU_CHILD: For new files (usually GIFs, JPEGs, or PNGs) that are embedded in the parent file.

  dwAssociation used in conjunction with dwOutputId can be used to segregate various types of files. For instance, the developer might want to place all GIFs in a sub-directory named GRAPHICS. Set by caller.
- dwOutputId: The type of the new file. This value is FI_XML_FLEXIONDOC_LATEST, FI_JPEGFIF, FI_GIF or FI_PNG.
- dwFlags: Reserved
- dwSpecType: IO specification type. See Data Access Common Functions for details about IO specifications.

  This member in conjunction with pSpec allows the developer to choose any location for the new file or even redirect its IO calls entirely. See Redirected IO for more details. 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 contain a proposed filename for the new file. When the SCCOPT_UNICODECALLBACKSTR option is set to TRUE, this filename is in Unicode. Otherwise, it is in single-byte characters. It is suggested, although not required, that this filename be used for the new file. Often the proposed filename will be referenced from within the output XML, so if the developer chooses a different one it may prevent consumers of the output from locating the files referenced from within the output.
- 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 EXURLFILEIOCALLBACKDATA / EXURLFILEIOCALLBACKDATAW Structures. Set by caller.

- pTemplateName: Not used in XML Export.

9.1.1 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.

9.2 EX_CALLBACK_ID_GRAPHICEXPORTFAILURE

This callback only occurs when an error is encountered exporting a graphic. It allows the OEM to customize their handling of this type of error. This callback does not occur for graphics exports that are successful. It also does not occur for graphics that cannot be converted due to the lack of an appropriate type of import filter. If the appropriate import filter is not present, EXOpenExport returns SCCERR_NOFILTER.

The pCommandOrInfoData field points to a structure of type EXGRAPHICEXPORTINFO:

```
typedef struct EXGRAPHICEXPORTINFOtag
{
   HIOFILE     hFile;
   VTLPDWORD   pXSize;
   VTLPDWORD   pYSize;
   VTDWORD     dwOutputId;
   SCCERR      ExportGraphicStatus;
   VTLPDWORD   pImageSize;
} EXGRAPHICEXPORTINFO;
```

- hFile: A handle to the current graphic output file. An OEM can substitute their own graphic by writing the desired graphic image to the beginning of the hFile (via an IOSEEK (hFile, IOSEEK_TOP, 0L), etc. The export function closes the file when control is returned from the callback. The contents of hFile on entry to the callback handler are unpredictable.
- pXSize/pYsize: Pointers to the dimensions of the image that would have been exported. An OEM can set and use these values to control the image size displayed by browsers. These dimensions are placed in the associated <img> tag.
- dwOutputId: The type of graphics file that was being created (FI_GIF, FI_JPEGFIF, or FI_PNG).
- ExportGraphicStatus: The error code from the operation that caused the graphic image conversion to fail.
- pImageSize: The maximum size for the image in bytes is filled in by HTML Export here (0 = no limit). If this callback is handled, on return the OEM should set this field to the size of the image the OEM created. This image should be no larger than the maximum size HTML Export entered into this variable.

Return Value

The callback handler should return SCCERR_NOTHANDLED unless the OEM has written an image to hFile in which case a value of SCCERR_OK should be returned.

9.3 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.

10 XML C/C++ Export Options

Options are parameters affecting the behavior of an export or transformation. This chapter presents C/C++ options available to the developer when using the XML Export engine.
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.

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.

The following types of options are covered:

- Character Mapping
- Output
- Input Handling
- Compression
- Graphics
- Callbacks
- XML
- File System

10.1 Character Mapping

This section pertains to character mapping options.

10.1.1 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. The possible character sets are listed in charsets.h.

When "extended test for text" is enabled (see 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 would be the value that takes effect.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Default

- CS_SYSTEMDEFAULT: Query the operating system.

Data

The data types are listed in charsets.h.

10.1.2 SCCOPT_UNMAPPABLECHAR

This option selects the character used when a character is not a valid Unicode character, or does not conform to the XML specification for valid characters. 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.

Handle Types

VTHDOC

Scope

Local

Data Type

VTWORD

Data

The Unicode value for the character to use.

Default

- 0xfffd

10.2 Output

This information pertains to output options.

10.2.1 SCCOPT_RENDERING_PREFER_OIT

Note:

This option is valid on 32-bit Linux (Red Hat and Suse), Linux x86-64, Solaris Sparc, IBM AIX 32-bit, and HP-UX RISC 32-bit platforms.

When this option is set to TRUE, the technology will attempt to use its internal graphics code to render fonts and graphics. When set to FALSE, the technology will render images using the operating system's native graphics subsystem (X11 on UNIX/ Linux platforms). Note that this option only works when at least one of the appropriate output solutions is present. For example, if the UNIX $DISPLAY variable does not point to a valid X Server, but the OSGD and/or WV_GD modules required for the Outside In output solution exist, Outside In will default to the Outside In rendering code. The option will fail if neither of these output solutions is present.

Note:

It is important for the system to be able to locate useable fonts when this option is set to TRUE. Only TrueType fonts (*.ttf or *.ttc files) are currently supported. To ensure that the system can find them, make sure that the environment variable GDFONTPATH includes one or more paths to these files. If the variable GDFONTPATH can't be found, the current directory is used. If fonts are called for and cannot be found, XML Export will exit with an error. Oracle does not provide fonts with any Outside In product.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTBOOL

Data

One of the following values:

- TRUE: Use the technology's internal graphics rendering code to produce bitmap output files whenever possible.

- FALSE: Use the operating system's native graphics subsystem.

Default

FALSE

10.3 Input Handling

This section pertains to input handling options.

10.3.1 SCCOPT_EXTRACTXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables the XMP feature, which does not interpret the XMP metadata, but passes it straight through without any interpretation. This option will be ignored if the SCCOPT_PARSEXMPMETADATA option is enabled.

Handle Types

VTHDOC

Scope

Local (was Global prior to release 8.2.2)

Data Type

VTBOOL

Data

- TRUE: This setting enables XMP extraction.
- FALSE: This setting disables XMP extraction.

Default

- FALSE

10.3.2 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.

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 selected specific plain-text character sets are no longer to be used. Instead, applications should use the SCCOPT_DEFAULTINPUTCHARSET option for such functionality.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Data

The high VTWORD of this value is reserved and should be set to 0, and the low VTWORD must have one of the following values:

- FI_TEXT: Unidentified file types will be treated as text files.
- FI_NONE: Outside In will not attempt to process files whose type cannot be identified. This will include text files. When this option is selected, an attempt to process a file of unidentified type will cause Outside In to return an error value of DAERR_FILTERNOTAVAIL (or SCCERR_NOFILTER).

Default

- FI_TEXT

10.3.3 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).

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Data

One of the following values:

- SCCUT_FI_NORMAL: This is the default value. When this is set, standard file identification behavior occurs.

- SCCUT_FI_EXTENDEDTEST: If set, the File Identification code will run an extended test on all files that are not identified.

Default

- SCCUT_FI_NORMAL

10.3.4 SCCOPT_FORMATFLAGS

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

- SCCOPT_FLAGS_ISODATETIMES: 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.
- SCCOPT_FLAGS_STRICTFILEACCESS: When an embedded file or URL can't be opened with the full path, OIT will sometimes try and open the referenced file from other locations, including the current directory. When this flag is set, it will prevent OIT from trying to open the file from any location other than the fully qualified path or URL.

Default

0: All flags turned off

10.3.5 SCCOPT_SYSTEMFLAGS

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

Handle Type

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

- SCCVW_SYSTEM_UNICODE: This flag causes the strings in SCCDATREENODE to be returned in Unicode.

Default

0

10.3.6 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.

Scope

Global

Data Type

VTBOOL

Data

- TRUE: Ignore validation of the password
- FALSE: Prompt for the password

Default

FALSE

10.3.7 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 NSF Support on Win x86-32 or Win x86-64 or NSF Support on Linux x86-32 or Solaris Sparc 32.

Handle Types

NULL

Scope

Global

Data Type

VTLPBYTE

Data

A path to the Lotus Notes directory.

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.

10.3.8 SCCOPT_PARSEXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables parsing of the XMP data into normal OIT document properties. Enabling this option may cause the loss of some regular data in premium graphics filters (such as Postscript), but won't affect most formats (such as PDF).

Handle Types

VTHDOC

Scope

Local

Data Type

VTBOOL

Data

- TRUE: This setting enables parsing XMP.
- FALSE: This setting disables parsing XMP.

Default

FALSE

10.3.9 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.

Handle Types

VTHDOC, NULL

Scope

Global

Data Type

VTDWORD

Data

- SCCUT_FILTER_STANDARD_BIDI
- SCCUT_FILTER_REORDERED_BIDI

Default

SCCUT_FILTER_STANDARD_BIDI

10.3.10 SCCOPT_PROCESS_OLE_EMBEDDINGS

Microsoft Powerpoint versions from 1997 through 2003 had the capability to embed OLE documents in the Powerpoint files. This option controls which embeddings are to be processed as native (OLE) documents and which are processed using the alternate graphic.

Note:

The Microsoft Powerpoint application sometimes does embed known Microsoft OLE embeddings (such as Visio, Project) as an "Unknown" type. To process these embeddings, the SCCOPT_PROCESS_OLEEMBED_ALL option is required. Post Office-2003 products such as Office 2007 embeddings also fall into this category.

Handle Types

VTHDOC, NULL

Scope

Global

Data Type

VTWORD

Data

- SCCOPT_PROCESS_OLEEMBED_ALL : Process all embeddings in the file
- SCCOPT_PROCESS_OLEEMBED_NONE : Process none of the embeddings in the file
- SCCOPT_PROCESS_OLEEMBED_STANDARD (default) : Process embeddings that are known standard embeddings. These include Office 2003 versions of Word, Excel, Visio etc.

Default

SCCOPT_PROCESS_OLEEMBED_STANDARD

10.3.11 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:**
>
> 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.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTLONG

Data

Integer parameter from -96 to 96, representing 15-minute offsets from GMT. To query the operating system for the time zone set on the machine, specify SCC_TIMEZONE_USENATIVE.

Default

- 0: GMT time

10.3.12 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.

Handle Type

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

- One or more of the following values OR-ed together:
- HTML_COND_COMMENT_NONE: Don't output any conditional comments. Note: setting any other flag will negate this.
- HTML_COND_COMMENT_IE5: include the IE 5 comments
- HTML_COND_COMMENT_IE6: include the IE 6 comments
- HTML_COND_COMMENT_IE7: include the IE 7 comments
- HTML_COND_COMMENT_IE8: include the IE 8 comments
- HTML_COND_COMMENT_IE9: include the IE 9 comments
- HTML_COND_COMMENT_ALL: include all conditional comments including the versions listed above and any other versions that might be in the HTML.

Default

HTML_COND_COMMENT_NONE

10.3.13 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

Data

- TRUE: Display the full path.
- FALSE: Do not display the path.

Default

FALSE

10.3.14 SCCOPT_STROKE_TEXT

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.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Default

TRUE

10.3.15 SCCOPT_PDF_FILTER_MAX_EMBEDDED_OBJECTS

PDF files sometimes have a very large number of embedded objects. This option allows the user to limit the number of embedded objects that are produced in a PDF file. Setting this option to 0 produces an unlimited number of embedded objects.

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

The maximum number of embedded objects to produce in PDF output.

Default

0

10.3.16 SCCOPT_PDF_FILTER_MAX_VECTOR_PATHS

PDF files sometimes have a very large number of vector paths. This option allows the user to limit the number of vector paths that are produced in a PDF file. Setting this option to 0 produces an unlimited amount of vector paths.

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

The maximum number of vector paths to produce in PDF output.

Default

0

10.3.17 SCCOPT_PDF_FILTER_WORD_DELIM_FRACTION

This option controls the spacing threshold in PDF input documents. Most PDF documents do not have an explicit character denoting a word break. The PDF filter calculates the distance between two characters to determine if they are part of the same word or if there should be a word break inserted. The space between characters is compared to the length of the space character in the current font multiplied by this fraction. If the space between characters is larger, then a word break character is inserted into the text stream. Otherwise, the characters are considered to be part of the same word and no word break is inserted.

Handle Types

NULL, VTHDOC

Scope

Local

Data Type

VTFLOAT

Data

A fraction representing the percentage of the space character used to trigger a word break. Valid values are 0<value<=2.

Default

0.85

10.3.18 SCCOPT_GENERATEEXCELREVISIONS

This option enables you to extract tracked changes from Excel. Extracted content shall include location (worksheet, row, column), author, date, and time. Note that Excel has an option to display the changes inline or on a different sheet. Either case should be extracted along with where the comments are displayed in the Excel file (inline or separate sheet).

Handle Types

VTHDOC

Scope

Global

Data Type

VTBOOL

Data

- TRUE: The setting enables generating Excel revision data
- FALSE: This setting disables generating Excel revision data

Default

FALSE

10.4 Compression

This section discusses compression options.

10.4.1 SCCOPT_FILTERJPG

This option can disable access to any files using JPEG compression, such as JPG graphic files or TIFF files using JPEG compression, or files with embedded JPEG graphics. Attempts to read or write such files when this option is enabled will fail and return the error SCCERR_UNSUPPORTEDCOMPRESSION if the entire file is JPEG compressed, and grey boxes for embedded JPEG-compressed graphics.

The following is a list of file types affected when this option is disabled:

- JPG files
- Postscript files containing JPG images
- PDFs containing JPEG images

Note that the setting for this option overrides the requested output graphic format when there is a conflict.

Handle Types

VTHDOC, HEXPORT

Scope

Global

Data Type

VTDWORD

Data

- SCCVW_FILTER_JPG_ENABLED: Allow access to files that use JPEG compression
- SCCVW_FILTER_JPG_DISABLED: Do not allow access to files that use JPEG compression

Default

SCCVW_FILTER_JPG_ENABLED

10.4.2 SCCOPT_FILTERLZW

This option can disable access to any files using Lempel-Ziv-Welch (LZW) compression, such as .GIF files, .ZIP files or self-extracting archive (.EXE) files containing "shrunk" files. Attempts to read or write such files when this option is enabled will fail and return the error SCCERR_UNSUPPORTEDCOMPRESSION if the entire file is LZW compressed, and grey boxes for embedded LZW-compressed graphics.

The following is a list of file types affected when this option is disabled:

- GIF files
- TIF files using LZW compression
- PDF files that use internal LZW compression
- TAZ and TAR archives containing files that are identified as `FI_UNIXCOMP`
- ZIP and self-extracting archive (.EXE) files containing "shrunk" files
- Postscript files using LZW compression

Note:

Although this option can disable access to files in ZIP or EXE archives stored using LZW compression, any files in such archives that were stored using any other form of compression will still be accessible. The setting for this option overrides the requested output graphic format when there is a conflict.

Handle Types

VTHDOC, HEXPORT

Scope

Global

Data Type

VTDWORD

Data

- SCCVW_FILTER_LZW_ENABLED: LZW compressed files will be read and written normally.
- SCCVW_FILTER_LZW_DISABLED: LZW compressed files will not be read or written.

Default

SCCVW_FILTER_LZW_ENABLED

10.5 Graphics

This information pertains to graphics options.

10.5.1 SCCOPT_ACCEPT_ALT_GRAPHICS

This option enables an optimization in XML Export's graphics output when exporting embedded graphics from an input document. When this option is set to TRUE and the input document contains embedded graphics that are already in one of our supported output formats, they will be copied to output files rather than converted to the selected output format specified by the SCCOPT_GRAPHIC_TYPE option.

For example, if this option is set to TRUE and the selected output graphics type is GIF, an input document's embedded JPEG graphic will be copied to a JPEG output file rather than being converted to the GIF format. The same behavior applies to all of XML Export's supported graphics output formats (currently GIF, JPEG, and PNG).

If this option is set to FALSE, all graphics output will be in the format specified by the SCCOPT_GRAPHIC_TYPE option.

Note:

When using this option, JPEG files will be transferred directly to their output file location, without being filtered. This presents the possibility that any JPEG viruses in the file can be transferred to that location, as well.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Data

- TRUE: FI_GIF, FI_JPEGFIF, and FI_PNG embeddings will be extracted, not converted. All other embeddings will be converted to the format specified by SCCOPT_GRAPHIC_TYPE. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.
- FALSE: All embeddings will be converted to the format specified by SCCOPT_GRAPHIC_TYPE. Embeddings that are already in that format will be extracted, not converted. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.

Default

FALSE

10.5.2 SCCOPT_GIF_INTERLACED

This option allows the developer to specify interlaced or non-interlaced GIF output. Interlaced GIFs are useful when graphics are to be downloaded over slow Internet connections. They allow the browser to begin to render a low-resolution view of the graphic quickly and then increase the quality of the image as it is received. There is no real penalty for using interlaced graphics.

This option is only valid if the SCCOPT_GRAPHIC_TYPE option is set to FI_GIF.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Data

One of the following values:

- TRUE: Produce interlaced GIFs.
- FALSE: Produce non-interlaced GIFs.

Default

TRUE

10.5.3 SCCOPT_GRAPHIC_HEIGHTLIMIT

This is an advanced option that casual users of this technology may safely ignore. It allows a hard limit to be set for how tall in pixels an exported graphic may be. Any images taller than this limit will be resized to match the limit. It should be noted that regardless whether the SCCOPT_GRAPHIC_WIDTHLIMIT option is set or not, any resized images will preserve their original aspect ratio.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

The maximum height of the output graphic in pixels. A value of zero is equivalent to SCCGRAPHIC_NOLIMIT, which causes this option to be ignored.

Default

- SCCGRAPHIC_NOLIMIT: No absolute height limit specified.

10.5.4 SCCOPT_GRAPHIC_OUTPUTDPI

This is an advanced option that casual users of this technology may safely ignore.

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 (SCCOPT_GRAPHIC_OUTPUTDPI is set to 50). In this case, the size of the resulting JPEG, GIF, or PNG 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.

Note:

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 will use the current screen resolution as the DPI setting for any other type of input file.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

The DPI to use when exporting graphic images. The maximum value allowed is SCCGRAPHIC_MAX_SANE_BITMAP_DPI, which is currently defined to be 2400 DPI.

Default

- SCCGRAPHIC_DEFAULT_OUTPUT_DPI: Currently defined to be 96 dots per inch.

10.5.5 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.

SCCOPT_GRAPHIC_SIZELIMIT takes precedence over all other options and settings that affect the size of a converted graphic.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

The total number of pixels in the output graphic. A value of zero ("0"), which is equivalent to SCCGRAPHIC_NOLIMIT, causes this option to be ignored.

Default

- SCCGRAPHIC_NOLIMIT: Option is turned off.

10.5.6 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.

Note:

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

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

One of the following values:

- SCCGRAPHIC_QUICKSIZING: Resize without antialiasing
- SCCGRAPHIC_SMOOTHSIZING: Resize using antialiasing
- SCCGRAPHIC_SMOOTHGRAYSCALESIZING: Resize using antialiasing for grayscale graphics only (no antialiasing for color graphics)

Default

SCCGRAPHIC_SMOOTHSIZING

10.5.7 SCCOPT_GRAPHIC_TYPE

This option allows the developer to specify the format of the graphics produced by the technology when it converts document embeddings.

When setting this option, remember that the JPEG file format does not support transparency.

Though the GIF file format supports transparency, it is limited to using only one of its 256 available colors to represent a transparent pixel ("index transparency").

PNG supports many types of transparency. The PNG files written by XML Export are created so that various levels of transparency are possible for each pixel. This is achieved through the implementation of an 8-bit "alpha channel."

There is a special optimization that XML Export can make when this option is set to FI_NONE. Some of the Outside In Viewer Technology's import filters can be optimized to ignore certain types of graphics. To take advantage of this optimization, the option must be set before EXOpenExport is called.

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.

Note:

The settings for options in Compression may force an override of the value for this option.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

One of the following values:

- FI_GIF: GIF graphics
- FI_JPEGFIF: JPEG graphics
- FI_PNG: PNG graphics
- FI_NONE: Graphic conversion will be turned off

Default

FI_JPEGFIF

10.5.8 SCCOPT_GRAPHIC_WIDTHLIMIT

This is an advanced option that casual users of this technology may safely ignore. It 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 SCCOPT_GRAPHIC_HEIGHTLIMIT option is set or not, any resized images will preserve their original aspect ratio.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

The maximum width of the output graphic in pixels. A value of zero is equivalent to SCCGRAPHIC_NOLIMIT, which causes this option to be ignored.

Default

- SCCGRAPHIC_NOLIMIT: No absolute width limit specified.

10.5.9 SCCOPT_JPEG_QUALITY

This option allows the developer to specify the lossyness of JPEG compression. The option is only valid if the SCCOPT_GRAPHIC_TYPE option is set to FI_JPEGFIF.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

A value from 1 to 100, with 100 being the highest quality but the least compression, and 1 being the lowest quality but the most compression.

Default

100

10.5.10 SCCOPT_RENDER_ENABLEALPHABLENDING

This option allows the user to enable alpha-channel blending (transparency) in rendering vector images when using an X-Windows output solution. This may improve fidelity on documents that use these transparent images, but will result in performance degradation. This option does not affect Microsoft Windows or Unix implementations where SCCOPT_RENDERING_PREFER_OIT is set to TRUE.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Default

False

10.6 Callbacks

This information pertains to callback options.

10.6.1 SCCOPT_EX_CALLBACKS

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

This option is used to disable callbacks being made from XML Export. Callbacks that are disabled will behave as if they were made and the developer had returned SCCERR_NOTHANDLED.

The option takes a VTDWORD field of flags. When the flag is set, the callback is enabled. By default, all callbacks are enabled. You can activate multiple callbacks by bitwise OR-ing them together. You can also disable multiple callbacks by bitwise &-ing the SCCEX_CALLBACKFLAG_ALLENABLED value with the one's complement of the corresponding callback flags. The following #defines are to be used for enabling the various callbacks:

Flag	Associated Callbacks
SCCEX_CALLBACKFLAG_CREATENEWFILE	EX_CALLBACK_ID_CREATENEWFILE
SCCEX_CALLBACKFLAG_NEWFILEINFO	EX_CALLBACK_ID_NEWFILEINFO

In addition, the following two special values are available:

- SCCEX_CALLBACKFLAG_ALLDISABLED: Disables the receipt of all callbacks. Additionally, bitwise OR-ing this value with one or more flags enables the corresponding callbacks.
- SCCEX_CALLBACKFLAG_ALLENABLED: Enables the receipt of all callbacks. Additionally, bitwise &-ing this value with the one's complement of one or more flags disables the corresponding callbacks. For example, SCCEX_CALLBACKFLAG_ALLENABLED& (~SCCEX_CALLBACKFLAG_CREATENEWFILE) disables the CREATENEWFILE callbacks, but enables all others.

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

One or more of the valid flags, bitwise OR-ed together

Default

- SCCEX_CALLBACKFLAG_ALLENABLED: All callbacks are available to the developer.

10.6.2 SCCOPT_EX_UNICODECALLBACKSTR

This option determines the format of strings used in the callback functions. For those structures that contain a field of type BYTE or LPBYTE, a comparable structure has been added which has a similar field of type WORD or LPWORD. These structures will have the same name as the original structure, with the addition of a "W" at the end.

When this option is set to TRUE, any time a callback uses a structure with a string, it will use the new structure. Also, any strings that the callback function returns will be expected to follow the same guidelines. If the option is set to FALSE, all callbacks will use single-byte character strings.

For example, if this option is set to TRUE, and the EX_CALLBACK_ID_CREATENEWFILE callback is called, the pExportData parameter to the callback will point to an EXURLFILEIOCALLBACKDATAW structure. If the option is set to FALSE, the pCommandOrInfoData parameter will point to an EXURLFILEIOCALLBACKDATA structure.

Note:

This option should be set before EXOpenExport is called.

Handle Types

VTHDOC

Scope

Local

Data Type

VTBOOL

Data

One of the following values:

- TRUE: Use Unicode strings in callbacks.
- FALSE: Do not use Unicode strings in callbacks.

Default

FALSE

10.7 XML

This information pertains to XML options.

10.7.1 SCCOPT_CCFLEX_FORMATOPTIONS

This option is a set of flags that can be set to affect the output.

Handle Types

VTHDOC

Scope

Local

Data Type

DWORD

Data

The following are the available flags for this option:

- CCFLEX_FORMATOPTIONS_DELIMITERS: Often, files have individual characters that are placed at specific draw locations. Consequently, the Flexiondoc converter produces individual `draw_text` characters without any indication of word boundaries. This flag forces the Flexiondoc converter to attempt to determine where words and lines end. The input filters indicate these positions by producing a `WORD_DELIMITER` for word endings, and a `DELIMITER` for line endings. These delimiters are passed along in the Flexiondoc output to assist the user in reconstructing words and lines.
- CCFLEX_FORMATOPTIONS_OPTIMIZESECTIONS: Use wp.section elements to delineate column references.
- CCFLEX_FORMATOPTIONS_FLATTENSTYLES: Flatten styles to eliminate the need to process the "based-on" attribute. By turning on this option, paragraph style should all be fully attributed. Character styles can't be fullly attributed, that is, they won't always be completely flattened.
- SCCOPT_FLAGS_ALLISODATETIMES: Use ISO 8601 formatting for all date and date/time values.
- CCFLEX_FORMATOPTIONS_PROCESSARCHIVESUBDOCS: Process all archive sub-objects and put the output in the main Flexiondoc output
- CCFLEX_FORMATOPTIONS_PROCESSATTACHMENTSUBDOCS: Process all attachments and put the output in the main Flexiondoc output

- CCFLEX_FORMATOPTIONS_PROCESSEMBEDDINGSUBDOCS: Process all embeddings and put the output in the main Flexiondoc output
- CCFLEX_FORMATOPTIONS_REMOVEFONTGROUPS: Replace font groups with references to individual fonts.
- CCFLEX_FORMATOPTIONS_INCLUDETEXTOFFSETS: Include text_offset attribute on tx.p and tx.r elements.
- CCFLEX_FORMATOPTIONS_SEPARATESTYLETABLES: Enabling this flag will cause the `style_tables` subtree to be streamed to a separate output unit. This item is deprecated.
- CCFLEX_FORMATOPTIONS_USEFULLFILEPATHS: Locators for externalized embeddings will contain full, absolute path names.
- CCFLEX_FORMATOPTIONS_BITMAPASBITMAP: dr.image objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.image.
- CCFLEX_FORMATOPTIONS_CHARTASBITMAP: ch.chart objects are converted to a graphic file and the resulting file is referenced by the locator child of the ch.chart.
- CCFLEX_FORMATOPTIONS_PRESENTATIONASBITMAP: pr.slide objects are converted to a graphic file and the resulting file is referenced by the locator child of the pr.slide.
- CCFLEX_FORMATOPTIONS_VECTORASBITMAP: dr.drawing objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.drawing.
- CCFLEX_FORMATOPTIONS_GENERATESYSTEMMETADATA: When this flag is set, system metadata will be genetated. This information is gathered through system calls and may adversely affect performance.

The following set of flags is useful if the caller is uninterested in certain kinds of elements. Setting these flags will eliminate entire categories of data from the conversion. Note: setting these flags may also remove part or all of the document properties or XMP data.

- CCFLEX_FORMATOPTIONS_NOBITMAPELEMENTS: Bitmap graphics are suppressed; no dr.image content will appear in the converted document.
- CCFLEX_FORMATOPTIONS_NOCHARTELEMENTS: Charts are suppressed; no ch.chart content will appear in the converted document.
- CCFLEX_FORMATOPTIONS_NOPRESENTATIONELEMENTS: Presentation slides are suppressed; no pr.slide content will appear in the converted document.
- CCFLEX_FORMATOPTIONS_NOVECTORELEMENTS: Vector drawings are suppressed; no dr.drawing content will appear in the converted document.

The following set of flags is useful when dealing with characters that can not be mapped to Unicode.

- CCFLEX_CHARMAPPING_DEFAULT (off): Default behavior: All text is mapped to Unicode, in tx.text elements.
- CCFLEX_CHARMAPPING_NOMAPPING: All text is left in the original character set, in tx.utext elements.
- CCFLEX_CHARMAPPING_MAPTEXT: Text is mapped to Unicode where possible, unmappable text is left in the original character set.

- CCFLEX_CHARMAPPING_BOTH: Both mapped and unmapped text is included as an alt element containing tx.text and tx.utext.

Default Value

All flags turned off, with the exception of: CCFLEX_FORMATOPTIONS_REMOVEFONTGROUPS.

10.7.2 SCCOPT_CCFLEX_INCLUDETEXTOFFSETS

Note:

This option is obsolete, having been superseded by the CCFLEX_FORMATOPTIONS_INCLUDETEXTOFFSETS flag in the SCCOPT_CCFLEX_FORMATOPTIONS option. However, it has been retained for backwards compatibility.

The value of this option is a Boolean that if set to TRUE will include offset information in the Flexiondoc output according to the schema. If the option is set to FALSE, no offset information is produced.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Default

FALSE

10.7.3 SCCOPT_CCFLEX_REMOVEFONTGROUPS

Note:

This option is obsolete, having been superseded by the CCFLEX_FORMATOPTIONS_REMOVEFONTGROUPS flag in the SCCOPT_CCFLEX_FORMATOPTIONS option. However, it has been retained for backwards compatibility.

Some word processing formats contain styles that reference font groups, forcing the user to interpret the correct font from that group by other means. If this option is set to TRUE, references to font groups in input documents are replaced with references to individual fonts.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTBOOL

Default

TRUE

10.7.4 SCCOPT_EXXML_DEF_METHOD

This option determines whether XML Export will reference the Flexiondoc schema, the Flexiondoc DTD, or no reference when generating output.

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

One of the following values:

- SCCEX_XML_XDM_DTD: Document Type Definition (DTD)
- SCCEX_XML_XDM_XSD: Extensible Schema Definition
- SCCEX_XML_XDM_NONE: No XML definition reference

Default

SCCEX_XML_XDM_NONE

10.7.5 SCCOPT_EXXML_DEF_REFERENCE

This option allows the developer to set a particular file as the XML definition reference.

If the SCCOPT_EXXML_DEF_METHOD xmlDefinitionMethod option is set to SCCEX_XML_XDM_XSD or SCCEX_XML_XDM_DTD, the value of this option will be used to reference the schema or DTD, respectively.

Handle Types

VTHDOC

Scope

Local

Data Type

Size (in bytes) of the data being passed, including a terminating `NULL`.

Data

The size of an array that holds WORD-sized characters terminated with a WORD-sized NULL (a UCS-2 string). The size passed is the total number of bytes that this UCS-2 string comprises. It includes in its size the bytes occupied by the terminating NULL.

Default

None

10.7.6 SCCOPT_EXXML_SUBSTREAMROOTS

Note:

As of the 8.1 release of Outside In XML Export, this option has been deprecated. Use the CCFLEX_FORMATOPTIONS_SEPARATESTYLETABLES flag in SCCOPT_CCFLEX_FORMATOPTIONS option instead.

This option selects the element which will be the root of a subtree of Flexiondoc output to be placed in a separate output document (a file or redirected IO stream). Currently, the only element supported for this option is the `style_tables` element.

When set to a non-empty or non-NULL value, this option specifies the subtree that should be exported to a separate document. This document, if it is a file, will be created in the same directory as the primary output document and will be named xxx.subtree.xml, where xxx.xml is the name of the primary document and subtree is the name of the exported element (for example, if output.xml is the primary output file, then the style_tables subtree would be exported to a file named output.style_tables.xml). When this option is set to an empty or NULL value, all elements will be placed in the primary output document.

An element specified in this option must include its namespace, followed by a comma, then the element name. Currently, the allowable values for this option are NULL, an empty string, or the following string:

```
http://www.outsideinsdk.com/xmlns/flexiondoc5_6,style_tables
```

Note:

This option will only work correctly if the SCCOPT_EXXML_DEF_METHOD option is set (to any value). If SCCOPT_EXXML_DEF_METHOD isn't set, the result will be an invalid output file.

Handle Types

VTHDOC, VTHEXPORT

Scope

Local

Data Type

VTDWORD

Data

The data for this option is a UCS-2 string (a NULL-terminated array of 16 bit Unicode characters). The data size should be specified as the length of the string in bytes (not characters), and should include the size of the terminating NULL.

Default

NULL

10.8 File System

This section discusses file system options.

10.8.1 SCCOPT_IO_BUFFERSIZE

This set of three options allows the user to adjust buffer sizes to tailor memory usage to the machine's ability. The numbers specified in these options are in kilobytes. These are advanced options that casual users of XML Export may ignore.

Handle Type

NULL, VTHDOC

Scope

Global

Data Type

SCCBUFFEROPTIONS Structure

Data

A buffer options structure

10.8.1.1 SCCBUFFEROPTIONS Structure

```
typedef struct SCCBUFFEROPTIONStag
{
   VTDWORD dwReadBufferSize;    /* size of the I/O Read buffer
                                in KB */
   VTDWORD dwMMapBufferSize;    /* maximum size for the I/O
                                Memory Map buffer in KB */
   VTDWORD dwTempBufferSize;    /* maximum size for the memory-
                                mapped temp files in KB */
   VTDWORD dwFlags;             /* use flags */
} SCCBUFFEROPTIONS, *PSCCBUFFEROPTIONS;
```

Parameters

- dwReadBufferSize: Used to define the number of bytes that will read from disk into memory at any given time. Once the buffer has data, further file reads will proceed within the buffer until the end of the buffer is reached, at which point the buffer will again be filled from the disk. This can lead to performance improvements in many file formats, regardless of the size of the document.
- dwMMapBufferSize: Used to define a maximum size that a document can be and use a memory-mapped I/O model. In this situation, the entire file is read from disk into memory and all further I/O is performed on the data in memory. This can lead to significantly improved performance, but note that either the entire file can be read into memory, or it cannot. If both of these buffers are set, then if the file is smaller than the dwMMapBufferSize, the entire file will be read into memory; if not, it will be read in blocks defined by the dwReadBufferSize.
- dwTempBufferSize: The maximum size that a temporary file can occupy in memory before being written to disk as a physical file. Storing temporary files in memory can boost performance on archives, files that have embedded objects or attachments. If set to 0, all temporary files will be written to disk.
- dwFlags
  - SCCBUFOPT_SET_READBUFSIZE 1
  - SCCBUFOPT_SET_MMAPBUFSIZE 2
  - SCCBUFOPT_SET_TEMPBUFSIZE 4

  To set any of the three buffer sizes, set the corresponding flag while calling dwSetOption.

Default

The default settings for these options are:

- #define SCCBUFOPT_DEFAULT_READBUFSIZE 2: A 2KB read buffer.
- #define SCCBUFOPT_DEFAULT_MMAPBUFSIZE 8192: An 8MB memory-map size.
- #define SCCBUFOPT_DEFAULT_TEMPBUFSIZE 2048: A 2MB temp-file limit.

Minimum and maximum sizes for each are:

- SCCBUFOPT_MIN_READBUFSIZE 1: Read one Kbyte at a time.
- SCCBUFOPT_MIN_MMAPBUFSIZE 0: Don't use memory-mapped input.

- SCCBUFOPT_MIN_TEMPBUFSIZE 0: Don't use memory temp files
- SCCBUFOPT_MAX_READBUFSIZE 0x003fffff, SCCBUFOPT_MAX_MMAPBUFSIZE 0x003fffff, SCCBUFOPT_MAX_TEMPBUFSIZE 0x003fffff: These maximums correspond to the largest file size possible under the 4GB `DWORD` limit.

10.8.2 SCCOPT_TEMPDIR

From time to time, the technology needs to create one or more temporary files. This option sets the directory to be used for those files.

It is recommended that this option be set as part of a system to clean up temporary files left behind in the event of abnormal program termination. By using this option with code to delete files older than a predefined time limit, the OEM can help to ensure that the number of temporary files does not grow without limit.

Note:

This option will be ignored if SCCOPT_REDIRECTTEMPFILE is set.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

SCCUTTEMPDIRSPEC structure

10.8.2.1 SCCUTTEMPDIRSPEC Structure

This structure is used in the SCCOPT_TEMPDIR option.

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

```
typedef struct SCCUTTEMPDIRSPEC
{
   VTDWORD   dwSize;
   VTDWORD   dwSpecType;
   VTBYTE    szTempDirName[SCCUT_FILENAMEMAX];
} SCCUTTEMPDIRSPEC,  * LPSCCUTTEMPDIRSPEC;
```

There is a limitation in the current release. dwSpecType describes the contents of szTempDirName. Together, dwSpecType and szTempDirName describe the location of the source file. The only dwSpecType values supported at this time are:

- IOTYPE_ANSIPATH: Windows only. szTempDirName 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. szTempDirName points to a NULL-terminated full path name using the Unicode character set and NTFS file name

conventions. Note that the length of the path name is limited to SCCUT_FILENAMEMAX bytes, or (SCCUT_FILENAMEMAX / 2) double byte Unicode characters.

- IOTYPE_UNIXPATH: UNIX platforms only. szTempDirName points to a NULL-terminated full path name using the system default character set and UNIX path conventions.

Specifically not supported at this time is IOTYPE_REDIRECT.

Parameters

- dwSize: Set to sizeof(SCCUTTEMPDIRSPEC).
- dwSpecType: IOTYPE_ANSIPATH, IOTYPE_UNICODEPATH, or IOTYPE_UNIXPATH
- szTempDirName: The path to the directory to use for the temporary files. Note that if all SCCUT_FILENAMEMAX bytes in the buffer are filled, there will not be space left for file names.

Default

The system default directory for temporary files. On UNIX systems, this is the value of environment variable $TMP. On Windows systems, it is the value of environment variable %TMP%.

10.8.3 SCCOPT_DOCUMENTMEMORYMODE

This option determines the maximum amount of memory that the chunker may use to store the document's data, from 4 MB to 1 GB. The more memory the chunker has available to it, the less often it needs to re-read data from the document.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Parameters

- SCCDOCUMENTMEMORYMODE_SMALLEST (4MB)
- SCCDOCUMENTMEMORYMODE_SMALL (16MB)
- SCCDOCUMENTMEMORYMODE_MEDIUM (64MB)
- SCCDOCUMENTMEMORYMODE_LARGE (256MB)
- SCCDOCUMENTMEMORYMODE_LARGEST (1 GB)

Default

SCCDOCUMENTMEMORYMODE_LARGE (256MB)

10.8.4 SCCOPT_REDIRECTTEMPFILE

This option is set when the developer wants to use redirected IO to completely take over responsibility for the low level IO calls of the temp file.

Handle Types

NULL, VTHDOC

Scope

Global (not persistent)

Data Type

VTLPVOID: pCallbackFunc

Function pointer of the redirect IO callback.

Redirect call back function:

```
typedef
{
     VTDWORD (* REDIRECTTEMPFILECALLBACKPROC)
     (HIOFILE *phFile,
     VTVOID *pSpec,
     VTDWORD dwFileFlags);
```

There is another option to handle the temp directory, SCCOPT_TEMPDIR. Only one of these two can be set by the developer. The SCCOPT_TEMPDIR option will be ignored if SCCOPT_REDIRECTTEMPFILE is set. These files may be safely deleted when the Close function is called.

Part III
Using the Java API

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

Part III contains the following chapters:

11
Introduction to the Java API

This chapter provides an introduction to the Java API for XML Export. 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.
The following topics are covered:

- Requirements
- Getting Started

11.1 Requirements

To use the API, the following set of modules and tools are required:

- Java JDK 6 or later
- The Outside In developer's redistributable modules for your product(s)
- 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.

11.2 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.

11.2.1 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*.)

11.2.2 Generate Code

Sample application code included with the SDK, OITSample, 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.

11.2.2.1 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();
```

11.2.2.2 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();
```

11.2.2.3 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.newArchiveNodeExporter(subdocId);
```

11.2.2.4 Set the Output Type

In this step, you specify the output format.

```
exporter.setDestinationFormat(FileFormat.FI_XML_FLEXIONDOC_LATEST);
```

11.2.2.5 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);
```

11.2.2.6 Run the Export

After all the previous steps are completed, you can produce the desired output.

```
exporter.export();
```

12 XML Export Java Classes

This chapter describes the XML Export Java classes.
The following classes are covered:

12.1 ArchiveNode Class

ArchiveNode provides information about an archive node. This is a read-only class where the technology fills in all the values.

Namespace

com.oracle.outsidein

Accessors

- boolean isFolder() - A value of true indicates that the record is an archive node.
- int getFileSize() - File size of the archive node
- java.util.Date getTime() - Time the archive node was created
- int getNodeNum() - Serial number of the archive node in the archive
- String getNodeName() - The name of the archive node

12.2 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.

Namespace

com.oracle.outsidein

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.

12.2.1 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.

12.2.1.1 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) throws IOException
```

- file: File object containing the full path to the new file
- url: A new URL that references the newly created file. This parameter can be null.

```
CreateNewFileResponse(SeekableByteChannel6 redirect, String url) throws IOException
```

- redirect: Object that will be written to as the destination of the transform
- 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.

12.2.2 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;
  }

  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);
  }
}
```

12.2.3 openFile

```
OpenFileResponse openFile(FileTypeFalue fileType, String fileName) throws
IOException
```

This callback is made any time a new file needs to be opened.

Parameters

- fileType: Type of file being requested to be opened
- fileName: Name of the file to be opened

Return Value

To take action in response to this method, return an OpenFileResponse object.

FileTypeValue Enumeration

This enumeration defines the type of file being requested to be opened. Its value may be one of the following:

- INPUT: File to be opened (path unknown)
- TEMPLATE: Template file to be opened
- PATH: Full file name of the file to be opened
- OTHER: Not used

12.2.3.1 OpenFileResponse Class

This is a class to define a new file or redirected I/O object in response to an openFile() callback.

Constructors

```
OpenFileResponse(File file)
```

- file: File object with full path to the new file

```
OpenFileResponse(SeekableByteChannel6 redirect)
```

- redirect: A redirected I/O object to which the file data will be written

12.2.4 createTempFile

```
CreateTempFileResponse createTempFile() throws IOException
```

This callback is made any time a new temporary file needs to be generated. This gives the developer the chance to handle the reading and writing of the temporary file.

Return Value

To take action in response to this notification, return a CreateTempFileResponse object with the temporary file information.

12.2.4.1 CreateTempFileResponseClass

This is a class to define a new redirected I/O object in response to a createTempFile() callback.

Constructors

```
CreateTempFileResponse(SeekableByteChannel6 redirect)
```

- redirect: A redirected I/O object to which the file data will be written and read from

12.3 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.

Namespace

com.oracle.outsidein

Methods

- **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.

- **newSubDocumentExporter**

  ```
  Exporter newSubDocumentExporter(
        int SubDocId,
        SubDocumentIdentifierTypeValue idType
  ) throws OutsideInException
  ```

  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

- **newSubObjectExporter**

  ```
  Exporter newSubObjectExporter(
        SubObjectTypeValue objType,
  ```

```
      int data1,
      int data2,
      int data3,
      int data4
) throws OutsideInException
```

Create a new Exporter for a subobject.

objType: Type of subobject

data1: Data identifying the subobject from SearchML

data2: Data identifying the subobject from SearchML

data3: Data identifying the subobject from SearchML

data4: Data identifying the subobject from SearchML

Returns: A new Exporter object for the subobject

SubObjectTypeValue: An enumeration to describe the type of SubObject to open.

– LinkedObject
– EmbeddedObject
– CompressedFile
– Attachment

- **newArchiveNodeExporter**

```
Exporter newArchiveNodeExporter(
      int dwRecordNum
) throws OutsideInException
```

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

- **export**

```
void export() throws OutsideInException
```

Perform the conversion.

- **setDestinationFile**

```
OptionsCache setDestinationFile(
      String filename
) throws OutsideInException
```

Set the location of the destination file

filename: Full path to the destination file

Returns: The updated options object

- **setExportTimeout**

```
OptionsCache setExportTimeout(int millisecondsTimeout)
```

This method sets the time that the export process should wait for a response from the Outside In export engine to complete the export of a document, setting an upper limit on the time that will elapse during a call to export(). If the specified length of time is reached before the export has completed, the export operation will be terminated and an OutsideInException will be thrown. If this option is not set, the default timeout is 5 minutes.

- **newLocalExporter**

  ```
  static Exporter newLocalExporter(Exporter source)
  ```

  This method creates and returns an instance of an Exporter object based on the source Exporter. All the options of source are copied to the new Exporter. The source and destination file information will not be copied.

12.3.1 Document Interface

All of the Outside In document-related methods are accessed through the Document Interface.

Namespace

com.oracle.outsidein

Methods

- **close**

  ```
  void close()
  ```

  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

- **getArchiveNode**

  ```
  ArchiveNode getArchiveNode(int nNodeNum) throws OutsideInException
  ```

Retrieves information about a record in an archive file. You may get the number of nodes in an archive using getArchiveNodeCount.

nNodeNum: The number of the record to retrieve information about. The first node is node 0.

Return Value: An ArchiveNode object with the information about the record

- **saveArchiveNode**

  ```
  void saveArchiveNode(
        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.

- **saveArchiveNode with Search Export Flags**

  ```
  void saveArchiveNode(
        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( String filename) throws OutsideInException
  ```

  Set the source document.

  filename: Full path of the source document

  Returns: The options cache object associated with this document

12.3.2 SeekableByteChannel6 Interface

Enables API users to handle I/O for the source and destination documents. Implement this interface to control I/O operations such as reading, writing, and seeking. This interface mimics the java.nio.channels.SeekableByteChannel interface which is only available in Java 7 and later. Note that SeekableByteChannel6 will be removed in favor of java.nio.channels.SeekableByteChannel if support for Java 6 is dropped in a future release of the Outside In Java API. Until then, this interface must be used if redirected I/O is required.

Namespace

com.oracle.outsidein

Methods

- **Get position**

  ```
  long position()
  ```

  Returns this channel's position.
- **Set position**

  ```
  SeekableByteChannel6 position(long newPosition)
  ```

  Sets this channel's position.
- **read**

  ```
  int read(java.nio.ByteBuffer dst)
  ```

  Reads a sequence of bytes from this channel into the given buffer. Bytes are read starting at this channel's current position, and then the position is updated with the number of bytes actually read.
- **size**

  ```
  long size()
  ```

  Returns the current size of the entity to which this channel is connected.
- **truncate**

  ```
  SeekableByteChannel6 truncate(long size)
  ```

  Truncates the entity, to which this channel is connected, to the given size. Never invoked by Outside In and may be implemented by just returning this.
- **write**

  ```
  int write(java.io.nio.ByteBuffer src)
  ```

  Writes a sequence of bytes to this channel from the given buffer. Bytes are written starting at this channel's current position. The entity to which the channel is connected is grown, if necessary, to accommodate the written bytes, and then the position is updated with the number of bytes actually written.
- **close**

  ```
  void close()
  ```

  Closes this channel. If this channel is already closed then invoking this method has no effect.
- **isOpen**

  ```
  boolean isOpen()
  ```

  Tells whether or not this channel is open.

12.3.3 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.

Namespace

com.oracle.outsidein.options

Methods

- OptionsCache setSourceFile(File file) throws OutsideInException

  Sets the source document to be opened.

  file: Full path to source file
- OptionsCache setSourceFile(SeekableByteChannel6 redirect) throws OutsideInException

  Sets an object that implements SeekableByteChannel6 to be used as the source document. Exporting a file using this method may have issues with files that require the original name of the file (examples: if the extension of the file is needed for identification purposes or if the name of a secondary file depends on the name/path of the original source file).

  redirect: Object implementing SeekableByteChannel6 to be used to read the source data containing the input file
- OptionsCache setSourceFile(SeekableByteChannel6 redirect, String filename) throws OutsideInException

  Sets an object that implements SeekableByteChannel6 to be used as the source document and provides information about the filename.

  redirect: Object implementing SeekableByteChannel6 to be used to read the source data containing the input file

  filename: A fully qualified path or file name that may be used to derive the extension of the file or name of a secondary file that is dependent on the name/path of the source file
- OptionsCache addSourceFile(File file) throws OutsideInException

  Sets the next source document file to be exported in sequence. This allows multiple documents to be exported to the same output destination.

  file: Full path to source file
- OptionsCache addSourceFile(SeekableByteChannel6 redirect)

  Set a redirected channel as the next source document to be exported to the original destination file. This method has the same limitations as the similar setSourceFile(SeekableByteChannel6 redirect) method.
- OptionsCache addSourceFile(SeekableByteChannel6 redirect, String Filename)

  Set a redirected channel as the next source document to be exported to the original destination file. The file name provided is used as in the method setSourceFile(SeekableByteChannel6 redirect, String Filename)
- 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 setDestinationFile(SeekableByteChannel6 redirect) throws OutsideInException

  Sets an object that implements SeekableByteChannel6 to be used as the destination document. An Exporter.export() operation will write the output data to the provided SeekableByteChannel6 object.

  redirect: Object implementing SeekableByteChannel6 to be used as the destination document written during an Exporter.export() operation

- 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.

- OptionsCache setOpenForNonSequentialAccess(boolean bOpenForNonSequentialAccess)

  Setting this option causes the technology to open archive files in a special mode that is only usable for non-sequential access of nodes.

  bOpenForNonSequentialAccess : If set to true would open the archive file in the special access mode. Note that turning this flag on a non-archive file will throw an exception at RunExport time.

12.3.3.1 AcceptAlternateGraphics

OIT Option ID: SCCOPT_ACCEPT_ALT_GRAPHICS

This option enables an optimization in XML Export's graphics output when exporting embedded graphics from an input document. When this option is set to TRUE and the input document contains embedded graphics that are already in one of our supported output formats, they will be copied to output files rather than converted to the selected output format specified by the GraphicType option.

For example, if this option is set to TRUE and the selected output graphics type is GIF, an input document's embedded JPEG graphic will be copied to a JPEG output file rather than being converted to the GIF format. The same behavior applies to all of XML Export's supported graphics output formats (currently GIF, JPEG, and PNG.)

If this option is set to FALSE, all graphics output will be in the format specified by the GraphicType option.

Note:

When using this option, JPEG files will be transferred directly to their output file location, without being filtered. This presents the possibility that any JPEG viruses in the file can be transferred to that location, as well.

Data Type

boolean

Data

- true: FI_GIF, FI_JPEGFIF, and FI_PNG embeddings will be extracted, not converted. All other embeddings will be converted to the format specified by GraphicType. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.
- false: All embeddings will be converted to the format specified by GraphicType. Embeddings that are already in that format will be extracted, not converted. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.

Default

false

12.3.3.2 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

SYSTEMDEFAULT

12.3.3.3 DocumentMemoryMode

OIT Option ID: SCCOPT_DOCUMENTMEMORYMODE

This option determines the maximum amount of memory that the chunker may use to store the document's data, from 4 MB to 1 GB. The more memory the chunker 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

SMALL: 2 - 16MB

12.3.3.4 EnableAlphaBlending

This option allows the user to enable alpha-channel blending (transparency) in rendering vector images. This is primarily useful to improve fidelity when rendering with a slower graphics engine, such as X-Windows over a network when performance is not an issue.

Data

Boolean

Default

False

12.3.3.5 ExtractXMPMetadata

OIT Option ID: SCCOPT_EXTRACTXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables the XMP feature, which does not interpret the XMP metadata, but passes it straight through without any interpretation. This option will be ignored if the ParseXMPMetadata option is enabled.

Data Type

boolean

Data

- true: This setting enables XMP extraction.
- false: This setting disables XMP extraction.

Default

- false

12.3.3.6 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

- 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

TEXT

12.3.3.7 GraphicHeight

OIT Option ID: SCCOPT_GRAPHIC_HEIGHT

This option defines the absolute height in pixels to which exported graphics will be resized. If this option is set and the GraphicWidth option is not, the width of the image will be calculated based on the aspect ratio of the source image. The developer should be aware that very large values for this option or GraphicWidth could produce images whose size exceeds available system memory, resulting in conversion failure.

If you are exporting a non-graphic file (word processing, spreadsheet or archive) and the settings for GraphicHeight and GraphicWidth do not match the aspect ratio of the original document, the exported image will have whitespace added so that the original file's aspect ratio is maintained.

The settings for the GraphicHeightLimit and GraphicWidth options can override the setting for GraphicHeight.

Data Type

long

12.3.3.8 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

long

12.3.3.9 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, 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 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; the default is 96 DPI.

Data Type

long

12.3.3.10 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

12.3.3.11 GraphicSizeMethod

OIT Option ID: 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

- QUICKSIZING
- SMOOTHSIZING
- SMOOTHGRAYSCALESIZING

12.3.3.12 GraphicWidth

OIT Option ID: SCCOPT_GRAPHIC_WIDTH

This option defines the absolute width in pixels to which exported graphics will be resized. If this option is set and the GraphicHeight option is not, the height of the image will be calculated based on the aspect ratio of the source image. The developer should be aware that very large values for this option or GraphicHeight could produce images whose size exceeds available system memory, resulting in conversion failure.

If you are exporting a non-graphic file (word processing, spreadsheet or archive) and the settings for GraphicHeight and GraphicWidth do not match the aspect ratio of the original document, the exported image will have whitespace added so that the original file's aspect ratio is maintained.

The settings for the GraphicHeightLimit and GraphicWidthLimit options can override the setting for GraphicWidth.

Data Type

long

12.3.3.13 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

12.3.3.14 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

- 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

12.3.3.15 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 should prompt for a password if it handles password-protected files.

Data Type

boolean

12.3.3.16 InterlacedGIFs

OIT Option ID: SCCOPT_GIF_INTERLACED

This option allows the developer to specify interlaced or non-interlaced GIF output. Interlaced GIFs are useful when graphics are to be downloaded over slow Internet connections. They allow the browser to begin to render a low-resolution view of the graphic quickly and then increase the quality of the image as it is received. There is no real penalty for using interlaced graphics.

This option is only valid if the dwOutputID parameter of the EXOpenExport function is set to FI_GIF.

Data Type

boolean

12.3.3.17 InternalRendering

Note:

This option is no longer relevant. Outside In no longer performs graphic rendering through X11 on Linux/Unix platforms.The internal rendering engine is available on all of these platforms. If this option is set, the results will always use the internal rendering engine regardless of the value of this option. The $GDFONTPATH environment variable must be set to specify where to reference fonts. On Windows systems, the Windows graphical rendering engine is always used.

12.3.3.18 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

12.3.3.19 JPEGQuality

OIT Option ID: SCCOPT_JPEG_QUALITY

This option allows the developer to specify the lossyness of JPEG compression. The option is only valid if the dwOutputID parameter of the EXOpenExport function is set to FI_JPEGFIF.

Data Type

long

Data

A value from 1 to 100, with 100 being the highest quality but the least compression, and 1 being the lowest quality but the most compression.

Default

100

12.3.3.20 LotusNotesDirectory

OIT Option ID: 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.

Type (Common): String

Data

A path to the Lotus Notes directory.

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.

12.3.3.21 OutputGraphicType

OIT Option ID: SCCOPT_GRAPHIC_TYPE

This option allows the developer to specify the format of the graphics produced by the technology.

- When setting this option, remember that the JPEG file format does not support transparency.
- Though the GIF file format supports transparency, it is limited to using only one of its 256 available colors to represent a transparent pixel ("index transparency").
- PNG supports many types of transparency. The PNG files written by HTML Export are created so that various levels of transparency are possible for each pixel. This is achieved through the implementation of an 8-bit "alpha channel".

There is a special optimization that HTML Export can make when this option is set to None. Some of the Outside In Viewer Technology's import filters can be optimized to ignore certain types of graphics.

Data Type

OutputGraphicTypeValue

OutputGraphicTypeValue Enumeration

These are the possible values for OutputGraphicType:

- GIF: Create GIF images
- JPEG: Create JPEG/JFIF images
- PNG: Create PNG images
- NONE: Turn off graphic conversions

Default

JPEG

12.3.3.22 ParseXMPMetadata

OIT Option ID: SCCOPT_PARSEXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables parsing of the XMP data into normal OIT document properties. Enabling this option may cause the loss of some regular data in premium graphics filters (such as Postscript), but won't affect most formats (such as PDF).

Data Type

boolean

Data

- true: This setting enables parsing XMP.
- false: This setting disables parsing XMP.

Default

false

12.3.3.23 PDFInputMaxEmbeddedObjects

This option allows the user to limit the number of embedded objects that are produced in a PDF file.

Data Type

long

Data

The maximum number of embedded objects to produce in PDF output. Setting this to 0 would produce an all embedded objects in the input document.

Default

0 – produce all objects.

12.3.3.24 PDFInputMaxVectorPaths

This option allows the user to limit the number of vector paths that are produced in a PDF file.

Data Type

long

Data

The maximum number of paths to produce in PDF output. Setting this to 0 would produce an all vector objects in the input document.

Default

0 – produce all vector objects.

12.3.3.25 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.

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.

12.3.3.26 PDFWordSpacingFactor

This option controls the spacing threshold in PDF input documents. Most PDF documents do not have an explicit character denoting a word break. The PDF filter calculates the distance between two characters to determine if they are part of the same word or if there should be a word break inserted. The space between characters is compared to the length of the space character in the current font multiplied by this fraction. If the space between characters is larger, then a word break character is inserted into the text stream. Otherwise, the characters are considered to be part of the same word and no word break is inserted.

Data Type

float

Data

A value representing the percentage of the space character used to trigger a word break. Valid values are positive values less than 2.

Default

0.85

12.3.3.27 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

12.3.3.28 ProcessOLEEmbeddingMode

OIT Option ID: SCCOPT_PROCESS_OLE_EMBEDDINGS

Microsoft Powerpoint versions from 1997 through 2003 had the capability to embed OLE documents in the Powerpoint files. This option controls which embeddings are to be processed as native (OLE) documents and which are processed using the alternate graphic.

Note:

The Microsoft Powerpoint application sometimes does embed known Microsoft OLE embeddings (such as Visio, Project) as an "Unknown" type. To process these embeddings, the ProcessOLEEmbedAll option is required. Post Office-2003 products such as Office 2007 embeddings also fall into this category.

Data

- STANDARD: Process embeddings that are known standard embeddings. These include Office 2003 versions of Word, Excel, Visio, etc.
- ALL: Process all embeddings in the file.
- NONE: Process none of the embeddings in the file.

Default

STANDARD

12.3.3.29 RenderEmbeddedFonts

This option allows you to disable the use of embedded fonts in PDF input files. If the option is set to true, the embedded fonts in the PDF input are used to render text; if the option is set to false, the embedded fonts are not used and the fallback is to use fonts available to Outside In to render text.

Data Type

boolean

Default

true

12.3.3.30 ShowArchiveFullPath

OIT Option ID: SCCOPT_ARCFULLPATH

This option causes the full path of a node to be returned in "GetArchiveNodeInfo" and "GetObjectInfo".

Data Type

boolean

Data

- true: Provide the full path.
- false: Do not provide the path.

Default

false

12.3.3.31 StrictFile

When an embedded file or URL can't be opened with the full path, OutsideIn will sometimes try and open the referenced file from other locations, including the current directory. When this option is set, it will prevent OutsideIn from trying to open the file from any location other than the fully qualified path or URL.

Data Type

boolean

Default

false

12.3.3.32 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:

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

Data

Integer parameter from -96 to 96, representing 15-minute offsets from GMT. To query the operating system for the time zone set on the machine, specify SCC_TIMEZONE_USENATIVE.

Default

- 0: GMT time

12.3.3.33 UnmappableCharacter

OIT Option ID: 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. It is left to the user to make sure that the selected replacement character is available in the output character set.

Data Type

int

Data

The Unicode value for the character to use.

Default

- 0x002a = "*"

12.3.3.34 XMLDefinitionReference

This option determines whether the converted file will reference a specified schema, DTD, or no reference when generating output.

Data Type

XMLReference

Data

A XMLReference object that defines the XML Definition Reference to be used.

Default

No reference defined

12.3.3.35 XXFormatOptions

This option is a set of flags that are specific to XML Export output files.

Data Type

EnumSet<XXFormatOptionValues>

XXFormatOptionValues Enumeration

The following set of flags:

- DELIMITERS: Often, files have individual characters that are placed at specific draw locations. Consequently, the Flexiondoc converter produces individual draw_text characters without any indication of word boundaries. This flag forces the Flexiondoc converter to attempt to determine where words and lines end. The input filters indicate these positions by producing a WORD_DELIMITER for word endings, and a DELIMITER for line endings. These delimiters are passed along in the Flexiondoc output to assist the user in reconstructing words and lines.
- OPTIMIZESECTIONS: Use wp.section elements to delineate column references.
- FLATTENSTYLES: Flatten styles to eliminate the need to process the "based-on=" attribute. By turning on this option, paragraph style should all be fully attributed. Character styles can't be fullly attributed, that is, they won't always be completely flattened.
- PROCESSARCHIVESUBDOCUMENTS: Process all archive sub-objects and put the output in the main Flexiondoc output

- PROCESSATTACHMENTSUBDOCUMENTS: Process all attachments and put the output in the main Flexiondoc output
- PROCESSEMBEDDINGSUBDOCUMENTS: Process all embeddings and put the output in the main Flexiondoc output
- REMOVEFONTGROUPS: Replace font groups with references to individual fonts.
- INCLUDETEXTOFFSETS: Include text_offset attribute on tx.p and tx.r elements.
- SEPARATESTYLETABLES: Enabling this flag will cause the style_tables subtree to be streamed to a separate output unit. This item is deprecated.
- USEFULLFILEPATHS: Locators for externalized embeddings will contain full, absolute path names.
- BITMAPASBITMAP: dr.image objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.image.
- CHARTASBITMAP: ch.chart objects are converted to a graphic file and the resulting file is referenced by the locator child of the ch.chart.
- PRESENTATIONASBITMAP: pr.slide objects are converted to a graphic file and the resulting file is referenced by the locator child of the pr.slide.
- VECTORASBITMAP: dr.drawing objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.drawing.
- GENERATESYSTEMMETADATA: When this flag is set, system metadata will be generated. This information is gathered through system calls and may adversely affect performance.
- NOBITMAPELEMENTS: Bitmap graphics are suppressed; no dr.image content will appear in the converted document.
- NOCHARTELEMENTS: Charts are suppressed; no ch.chart content will appear in the converted document.
- NOPRESENTATIONELEMENTS: Presentation slides are suppressed; no pr.slide content will appear in the converted document.
- NOVECTORELEMENTS: Vector drawings are suppressed; no dr.drawing content will appear in the converted document.
- These next four flags are mutually exclusive:
  - DEFAULTCHARACTERMAPPING: Default behavior: All text is mapped to Unicode, in tx.text elements.
  - NOCHARARACTERMAPPING: All text is left in the original character set, in tx.utext elements.
  - MAPTEXT: Text is mapped to Unicode where possible, unmappable text is left in the original character set.
  - MAPPEDANDUNMAPPEDCHARACTERS: Both mapped and unmapped text is included as an alt element containing tx.text and tx.utext.

Default

REMOVEFONTGROUPS

12.4 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.

Namespace

com.oracle.outsidein

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 that 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 sub documents 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 sub-document was not converted
- TypeThreeFont: A type 3 font was encountered.
- UnsupportedShading: An unsupported shading pattern was encountered.
- InvalidHTML: An HTML parse error, as defined by the W3C, was encountered.

12.5 FileFormat Class

This class defines the identifiers for file formats.

Namespace

com.oracle.outsidein

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.

12.6 ObjectInfo Class

ObjectInfo provides all the information available about the OIT Object. This is a read-only class where the technology fills in all the values.

Namespace

com.oracle.outsidein.options

Accessors

- ObjectInfo.CompressionValues getCompression() - the type of compression used to store the object, if known.
- EnumSet<ObjectInfo.ObjectInfoFlagValues> getFlags() - flags indicating attributes of the object.
- FileFormat getFormatId() - the format Identifier of the object.
- String getName() - name of the object.

ObjectInfoFlags Enumeration

Bit fields to describe information about an object.

- PARTIALFILE: Object would not normally exist outside the source document
- PROTECTEDFILE: Object is encrypted or password protected
- UNSUPPORTEDCOMPRESSION: Object uses an unsupported compression mechanism

- DRMFILE: Object uses Digital Rights Management protection
- UNIDENTIFIEDFILE: Object is extracted, but can not successfully identified
- LINKTOFILE: Object links to file, it can not be extracted
- ENCRYPTEDFILE: Object is encrypted and can be decrypted with the known password

12.7 Option Interface

The Option Interface provides the methods and properties to retrieve information about an Outside In Option.

Namespace

Outside In

Properties

- Name — Name of the option
- Description — Description of the option
- DataType — The type of the option value
- SupportingProducts — The list of products that support this option

Method

```
void Set(OptionsCache exporter, Object objValue);
```

This method sets the option to the exporter object.

- exporter — The exporter option
- objValue — Value of the option

Note:

If the type of objValue cannot be converted to the data type the option is expecting, an OutsideInCastException is thrown.

OutsideInProducts Enumeration

- HTMLExport — Outside In HTML Export
- ImageExport — Outside In Image Export
- PDFExport — Outside In PDF Export
- SearchExport — Outside In Search Export
- WebViewExport — Outside In Web View Export
- XMLExport — Outside In XML Export
- AllExports — All Outside In export products

12.8 OutsideIn Class

This is a utility class that creates an instance of an Exporter object on request.

Namespace

com.oracle.outsidein

Methods

```
static Exporter newLocalExporter()
```

This method creates an instance of an Exporter object. It returns a newly created Exporter object.

```
static Exporter newLocalExporter(Exporter source)
```

This method creates and returns an instance of an Exporter object based on the source Exporter. All the options of source are copied to the new Exporter. The source and destination file information will not be copied.

12.9 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.

Namespace

com.oracle.outsidein

12.10 XMLReference Class

The XMLReference class is a data class used to define the XML definition reference to be used.

Namespace

com.oracle.outsidein.options

Constructors

```
XMLReference()
```

Create an instance of a XMLReference object using No XML definition reference

```
XMLReference(XMLReference.ReferenceMethodValue, String)
```

Create an instance of a XMLReference object to provide a DTD/XSD

ReferenceMethodValue Enumeration

This enumeration is used to set whether Export will reference a schema, a DTD, or no reference when generating output.

- DTD: Document Type Definition (DTD)
- XSD: Extensible Schema Definition
- NONE: No definition reference

Part IV
Using the .NET API

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

Part IV contains the following chapters:

13

Introduction to the .NET API

This chapter is an introduction to the .NET API for XML Export. 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.

The following topics are covered:

- Requirements
- Getting Started

13.1 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.

13.2 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.

13.2.1 Configuring your Environment

To setup the environment to create a .NET application, you would need to add references to all the libraries. In order to use the Outside In components in your application, the following component should be referenced: outsidein.dll. (This can be done by using the Add Reference dialog box in Visual Studio.)

13.2.2 Generate Code

The sample application included with the SDK, OITsample, 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.

13.2.2.1 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();
```

13.2.2.2 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();
```

13.2.2.3 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);
```

13.2.2.4 Set the Output Type

In this step, you specify the output format.

```
exporter.SetDestinationFormat(FileFormat.FI_HTML5);
```

13.2.2.5 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 next section

}

CallbackHandler callback = new CallbackHandler();

exporter.SetCallbackHandler(callback);
```

13.2.2.6 Run the Export

After all the previous steps are completed, you can produce the desired output.

```
exporter.Export();
```

13.2.3 Redirected I/O Support in .NET

Support for redirected I/O is supported through .NET Streams. Streams that are readable and seekable can be used as input files, while streams that are readable, writable and seekable can be used for output.

Using streams is very similar to using standard I/O in the .NET API. To use streams, the stream object is passed as a parameter to the "SetSourceFile" or "SetDestinationFile". When using Output streams, handling callbacks is mandatory when secondary files are expected to be generated.

14 XML Export .NET Classes

This chapter describes the XML Export .NET classes.
The following classes are covered:

- ArchiveNode Class
- Callback Class
- Exporter Interface
- ExportStatus Class
- FileFormat Class
- ObjectInfo Class
- Option Interface
- OutsideIn Class
- OutsideInException Class
- XMLReference Class

14.1 ArchiveNode Class

ArchiveNode provides information about an archive node. This is a read-only class where the technology fills in all the values.

Namespace

OutsideIn

Properties

- IsDirectory (Boolean) A value of true indicates that the record is an archive node.
- FileSize (Int32) File size of the archive node
- NodeTime (Int32) Time the archive node was created
- NodeNum (Int32) Serial number of the archive node in the archive
- NodeName (String) The name of the archive node

14.2 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.

Namespace

OutsideIn

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 three methods: OpenFile, CreateNewFile and NewFileInfo.

14.2.1 OpenFile

```
OpenFileResponse OpenFile(
      FileTypeValue fileType,
      string fileName
)
```

This callback is made any time a new file needs to be opened.

Parameters

- fileType: Type of file being requested to be opened.
- filename: Name of the file to be open

Return Value

To take action in response to this method, return an OpenFileResponse object.

FileTypeValue Enumeration

This enumeration defines the type of file being requested to be opened. Its value may be one of the following:

- Input: File to be opened (path unknown)
- Template: Template file to be opened
- Path: Full file name of the file to be opened.
- Other: Not used.

14.2.1.1 OpenFileResponse Class

This is a class to define a new file or stream object in response to an OpenFile callback.

Constructor

```
OpenFileResponse(FileInfo file)
```

File: File object with full path to the new file.

```
OpenFileResponse(Stream file)
```

File: A stream to which the file data will be written.

14.2.2 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.

14.2.2.1 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(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.

14.2.3 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

14.2.4 CreateTempFile

```
CreateTempFileResponse CreateTempFile()
```

This callback is made any time a new temporary file needs to be generated. This gives the developer the chance to handle the reading and writing of the temporary file.

Return Value

To take action in response to this notification, return a CreateTempFileResponse object with the temporary file information.

14.2.4.1 CreateTempFileResponse Class

This is a class to define a new file or stream object in response to an CreateTempFile callback.

Constructor

```
CreateTempFileResponse (Stream file)
```

File: A stream to which the file data will be written and read from.

14.3 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.

Namespace

OutsideIn

Methods

- **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.

- **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.

  - IDTYPE_XX: Subdocument to be opened is based on output of XML Export (SubdocId is the value of the object_id attribute of a locator element.)
  - IDTYPE_ATTACHMENT_LOCATOR: Subdocument to be opened is based on the locator value provided by the one of the Export SDKs.
  - IDTYPE_ATTACHMENT_INDEX: 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

- **NewSubObjectExporter**

  ```
  Exporter NewSubObjectExporter(
        SubObjectTypeValue objType,
        uint data1,
        uint data2,
        uint data3,
        uint data4
  )
  ```

  Create a new Exporter for a subobject.

  objType: Type of subobject

  data1: Data identifying the subobject from SearchML

  data2: Data identifying the subobject from SearchML

  data3: Data identifying the subobject from SearchML

  data4: Data identifying the subobject from SearchML

  Returns: A new Exporter object for the subobject

SubObjectTypeValue: An enumeration to describe the type of SubObject to open.

- LinkedObject
- EmbeddedObject
- CompressedFile
- Attachment

- **NewArchiveNodeExporter**

```
Exporter NewArchiveNodeExporter(
      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

- **Export**

```
void Export()
```

  Perform the conversion and close the Export process.

- **SetExportTemplate**

```
SetExportTemplate(FileInfo template)
```

  This method sets the template file to be used for export.

  template: A FileInfo object representing the template to be used for export.

- **SetExportTimeout**

```
OptionsCache SetExportTimeout(int millisecondsTimeout);
```

  This method sets the time that the export process should wait for a response from the Outside In export engine to complete the export of a document, setting an upper limit on the time that will elapse during a call to Export(). If the specified length of time or the default timeout amount is reached before the export has completed, the export operation is terminated and an OutsideInException is thrown. If this option is not set, the default timeout is 5 minutes.

- **Close**

```
Close()
```

  This function closes the current Export process.

- **NewLocalExporter**

```
static Exporter NewLocalExporter(Exporter source)
```

  This method creates and returns an instance of an Exporter object based on the source Exporter. All the options of source are copied to the new Exporter. The source and destination file information will not be copied.

14.3.1 Document Interface

All of the Outside In document-related methods are accessed through the Document Interface.

Namespace

OutsideIn

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
- **GetArchiveNode**

  ```
  ArchiveNode GetArchiveNode(Int32 nNodeNum)
  ```

  Retrieves information about a record in an archive file. You may get the number of nodes in an archive using getArchiveNodeCount.

  nNodeNum: The number of the record to retrieve information about. The first node is node 0.

  Return Value: An ArchiveNode object with the information about the record
- **SaveArchiveNode**

  ```
  void SaveArchiveNode(
        Int32 nNodeNum,
        FileInfo fileinfo)
  void SaveArchiveNode(
  ```

```
      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

- **SaveArchiveNode with ArchiveNode**

  ```
  void SaveArchiveNode(
        ArchiveNode arcNode,
        FileInfo fileinfo)
  void SaveRecord(
        ArchiveNode arcNode,
        string strFileName)
  ```

  Extracts a record in an archive file to disk.

  arcNode: An ArchiveNode object retrieved from GetArchiveNodeInfo with information about the node to extract

  strFileNameType/fileinfo: Full path of the destination file to which the file will be extracted

- **SaveArchiveNode with Search Export Flags**

  ```
  void SaveArchiveNode(
        uint flags,
        uint params1,
        uint params2,
        FileInfo fileinfo)
  void SaveArchiveNode(
        uint flags,
        uint params1,
        uint params2,
        string strFileName)
  ```

  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

  strFileNameType/fileinfo: Full path of the destination file to which the file will be extracted

14.3.2 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.

Namespace

OutsideIn.Options

Methods

- OptionsCache SetSourceFile(FileInfo file)

  Sets the source document to be opened.

  file: Full path to source file

- OptionsCache SetSourceFile(string filename)

  Set the source document.

  filename: Full path of the source document

  Returns: The options cache object associated with this document

- OptionsCache AddSourceFile(FileInfo file)

  Sets the next source document file to be exported in sequence. This allows multiple documents to be exported to the same output destination.

  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 SetDestinationFile(string filename)

  Set the location of the destination file.

  filename: Full path to the destination file

  returns: The updated options object

- 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.

- OptionsCache SetOpenForNonSequentialAccess(bool bOpenForNonSequentialAccess)

  Setting this option causes the technology to open archive files in a special mode that is only usable for non-sequential access of nodes.

  bOpenForNonSequentialAccess : If set to true would open the archive file in the special access mode. Note that turning this flag on a non-archive file will throw an exception at RunExport time.

- OptionsCache SetSourceFile(Stream file)

  Set an input stream as the source document. Exporting a file using this method may have issues with files that require the original name of the file (example: extension of the file for identification purposes or name of a secondary file dependent on the name/path of this file).

- OptionsCache SetSourceFile(Stream file, String Filename)

  Set an input stream as the source document and provide information about the filename (fully qualified path or file name that may be used to derive the extension of the file or name of a secondary file dependent on the name/path of this file).

- OptionsCache SetNextSourceFile(Stream file)

  Set an input stream as the next source document to be exported to the original destination file. This method has the same limitations as the similar SetSourceFile(Stream file) method.

- OptionsCache SetNextSourceFile(Stream file, String Filename)

  Set an input stream as the next source document to be exported to the original destination file. The file name provided is used as in the method SetSourceFile(Stream file, String Filename)

- OptionsCache SetNextSourceFile(FileInfo file)

  Set an input stream as the next source document to be exported to the original destination file.

- OptionsCache SetDestinationFile(Stream file)

  Set an output stream as the destination for an export.

14.3.2.1 AcceptAlternateGraphics

OIT Option ID: SCCOPT_ACCEPT_ALT_GRAPHICS

This option enables an optimization in XML Export's graphics output when exporting embedded graphics from an input document. When this option is set to true and the input document contains embedded graphics that are already in one of our supported output formats, they will be copied to output files rather than converted to the selected output format specified by the GraphicType option.

For example, if this option is set to true and the selected output graphics type is GIF, an input document's embedded JPEG graphic will be copied to a JPEG output file rather than being converted to the GIF format. The same behavior applies to all of XML Export's supported graphics output formats (currently GIF, JPEG, and PNG.)

If this option is set to false, all graphics output will be in the format specified by the GraphicType option.

Note:

When using this option, JPEG files will be transferred directly to their output file location, without being filtered. This presents the possibility that any JPEG viruses in the file can be transferred to that location, as well.

Data Type

bool

Data

- true: FI_GIF, FI_JPEGFIF, and FI_PNG embeddings will be extracted, not converted. All other embeddings will be converted to the format specified by GraphicType. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.
- false: All embeddings will be converted to the format specified by GraphicType. Embeddings that are already in that format will be extracted, not converted. If graphicType is set to FI_NONE, no embeddings will be extracted or converted.

Default

false

14.3.2.2 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

SystemDefault

14.3.2.3 DocumentMemoryMode

OIT Option ID: SCCOPT_DOCUMENTMEMORYMODE

This option determines the maximum amount of memory that the chunker may use to store the document's data, from 4 MB to 1 GB. The more memory the chunker 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

SMALL: 2 - 16MB

14.3.2.4 ExtractXMPMetadata

OIT Option ID: SCCOPT_EXTRACTXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables the XMP feature, which does not interpret the XMP metadata, but passes it straight through without any interpretation. This option will be ignored if the ParseXMPMetadata option is enabled.

Data Type

bool

Data

- true: This setting enables XMP extraction.
- false: This setting disables XMP extraction.

Default

- false

14.3.2.5 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

- 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

Text

14.3.2.6 GraphicHeight

OIT Option ID: SCCOPT_GRAPHIC_HEIGHT

This option defines the absolute height in pixels to which exported graphics will be resized. If this option is set and the GraphicWidth option is not, the width of the image will be calculated based on the aspect ratio of the source image. The developer should be aware that very large values for this option or GraphicWidth could produce images whose size exceeds available system memory, resulting in conversion failure.

If you are exporting a non-graphic file (word processing, spreadsheet or archive) and the settings for GraphicHeight and GraphicWidth do not match the aspect ratio of the original document, the exported image will have whitespace added so that the original file's aspect ratio is maintained.

The settings for the GraphicHeightLimit and GraphicWidth options can override the setting for GraphicHeight.

Data Type

Int32

14.3.2.7 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

Int32

14.3.2.8 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, 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 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; the default is 96 DPI.

Data Type

Int32

14.3.2.9 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

Int32

14.3.2.10 GraphicSizeMethod

OIT Option ID: 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

- GRAPHICSIZEMETHOD_VALUES

14.3.2.11 GraphicWidth

OIT Option ID: SCCOPT_GRAPHIC_WIDTH

This option defines the absolute width in pixels to which exported graphics will be resized. If this option is set and the GraphicHeight option is not, the height of the image will be calculated based on the aspect ratio of the source image. The developer should be aware that very large values for this option or GraphicHeight could produce images whose size exceeds available system memory, resulting in conversion failure.

If you are exporting a non-graphic file (word processing, spreadsheet or archive) and the settings for GraphicHeight and GraphicWidth do not match the aspect ratio of the original document, the exported image will have whitespace added so that the original file's aspect ratio is maintained.

The settings for the GraphicHeightLimit and GraphicWidthLimit options can override the setting for GraphicWidth.

Data Type

Int32

14.3.2.12 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

Int32

14.3.2.13 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

- 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

14.3.2.14 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 should prompt for a password if it handles password-protected files.

Data Type

bool

14.3.2.15 InterlacedGIFs

OIT Option ID: SCCOPT_GIF_INTERLACED

This option allows the developer to specify interlaced or non-interlaced GIF output. Interlaced GIFs are useful when graphics are to be downloaded over slow Internet connections. They allow the browser to begin to render a low-resolution view of the graphic quickly and then increase the quality of the image as it is received. There is no real penalty for using interlaced graphics.

This option is only valid if the dwOutputID parameter of the EXOpenExport function is set to FI_GIF.

Data Type

bool

14.3.2.16 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

14.3.2.17 JPEGQuality

OIT Option ID: SCCOPT_JPEG_QUALITY

This option allows the developer to specify the lossyness of JPEG compression. The option is only valid if the dwOutputID parameter of the EXOpenExport function is set to FI_JPEGFIF.

Data Type

Int32

Data

A value from 1 to 100, with 100 being the highest quality but the least compression, and 1 being the lowest quality but the most compression.

Default

100

14.3.2.18 LotusNotesDirectory

OIT Option ID: 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.

Data

A path to the Lotus Notes directory.

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.

14.3.2.19 OutputGraphicType

OIT Option ID: SCCOPT_GRAPHIC_TYPE

This option allows the developer to specify the format of the graphics produced by the technology.

- When setting this option, remember that the JPEG file format does not support transparency.
- Though the GIF file format supports transparency, it is limited to using only one of its 256 available colors to represent a transparent pixel ("index transparency").
- PNG supports many types of transparency. The PNG files written by HTML Export are created so that various levels of transparency are possible for each pixel. This is achieved through the implementation of an 8-bit "alpha channel".

There is a special optimization that HTML Export can make when this option is set to None. Some of the Outside In Viewer Technology's import filters can be optimized to ignore certain types of graphics.

Data Type

OutputGraphicTypeValue

OutputGraphicTypeValue Enumeration

These are the possible values for OutputGraphicType:

- GIF: Create GIF images
- JPEG: Create JPEG/JFIF images
- PNG: Create PNG images
- NONE: Turn off graphic conversions

Default

JPEG

14.3.2.20 ParseXMPMetadata

OIT Option ID: SCCOPT_PARSEXMPMETADATA

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. This option enables parsing of the XMP data into normal OIT document properties. Enabling this option may cause the loss of some regular data in premium graphics filters (such as Postscript), but won't affect most formats (such as PDF).

Data Type

bool

Data

- true: This setting enables parsing XMP.
- false: This setting disables parsing XMP.

Default

false

14.3.2.21 PDFInputMaxEmbeddedObjects

This option allows the user to limit the number of embedded objects that are produced in a PDF file.

Data Type

UInt32

Data

The maximum number of embedded objects to produce in PDF output. Setting this to 0 would produce an all embedded objects in the input document.

Default

0 – produce all objects.

14.3.2.22 PDFInputMaxVectorPaths

This option allows the user to limit the number of vector paths that are produced in a PDF file.

Data

The maximum number of paths to produce in PDF output. Setting this to 0 would produce an all vector objects in the input document.

Default

0 – produce all vector objects.

14.3.2.23 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.

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.

14.3.2.24 PDFWordSpacingFactor

This option controls the spacing threshold in PDF input documents. Most PDF documents do not have an explicit character denoting a word break. The PDF filter calculates the distance between two characters to determine if they are part of the same word or if there should be a word break inserted. The space between characters is compared to the length of the space character in the current font multiplied by this fraction. If the space between characters is larger, then a word break character is inserted into the text stream. Otherwise, the characters are considered to be part of the same word and no word break is inserted.

Data Type

float

Data

A value representing the percentage of the space character used to trigger a word break. Valid values are positive values less than 2.

Default

0.85

14.3.2.25 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

14.3.2.26 ProcessOLEEmbeddingMode

OIT Option ID: SCCOPT_PROCESS_OLE_EMBEDDINGS

Microsoft Powerpoint versions from 1997 through 2003 had the capability to embed OLE documents in the Powerpoint files. This option controls which embeddings are to be processed as native (OLE) documents and which are processed using the alternate graphic.

Note:

The Microsoft Powerpoint application sometimes does embed known Microsoft OLE embeddings (such as Visio, Project) as an "Unknown" type. To process these embeddings, the ProcessOLEEmbedAll option is required. Post Office-2003 products such as Office 2007 embeddings also fall into this category.

Data

- Standard: Process embeddings that are known standard embeddings. These include Office 2003 versions of Word, Excel, Visio, etc.
- All: Process all embeddings in the file.
- None: Process none of the embeddings in the file.

Default

Standard

14.3.2.27 RenderEmbeddedFonts

This option allows you to disable the use of embedded fonts in PDF input files. If the option is set to true, the embedded fonts in the PDF input are used to render text; if the option is set to false, the embedded fonts are not used and the fallback is to use fonts available to Outside In to render text.

Data Type

bool

Default

true

14.3.2.28 ShowArchiveFullPath

OIT Option ID: SCCOPT_ARCFULLPATH

This option causes the full path of a node to be returned in "GetArchiveNodeInfo" and "GetObjectInfo".

Data Type

bool

Data

- true: Provide the full path.
- false: Do not provide the path.

Default

false

14.3.2.29 StrictFile

When an embedded file or URL can't be opened with the full path, OutsideIn will sometimes try and open the referenced file from other locations, including the current directory. When this option is set, it will prevent OutsideIn from trying to open the file from any location other than the fully qualified path or URL.

Data Type

bool

Default

false

14.3.2.30 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:**
>
> 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

Data

Integer parameter from -96 to 96, representing 15-minute offsets from GMT. To query the operating system for the time zone set on the machine, specify SCC_TIMEZONE_USENATIVE.

Default

- 0: GMT time

14.3.2.31 UnmappableCharacter

OIT Option ID: 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. It is left to the user to make sure that the selected replacement character is available in the output character set.

Data Type

UShort

Data

The Unicode value for the character to use.

Default

- 0x002a = "*"

14.3.2.32 XMLDefinitionReference

This option determines whether the converted file will reference a specified schema, DTD, or no reference when generating output.

Data Type

XMLReference

Data

A XMLReference object that defines the XML Definition Reference to be used.

Default

No reference defined

14.3.2.33 XXFormatOptions

This option is a set of flags that are specific to XML Export output files.

Data Type

XXFormatOptionValues

XXFormatOptionValues Enumeration

The following set of flags:

- Delimiters: Often, files have individual characters that are placed at specific draw locations. Consequently, the Flexiondoc converter produces individual draw_text characters without any indication of word boundaries. This flag forces the

Flexiondoc converter to attempt to determine where words and lines end. The input filters indicate these positions by producing a WORD_DELIMITER for word endings, and a DELIMITER for line endings. These delimiters are passed along in the Flexiondoc output to assist the user in reconstructing words and lines.

- OptimizeSections: Use wp.section elements to delineate column references.
- FlattenStyles: Flatten styles to eliminate the need to process the "based-on=" attribute. By turning on this option, paragraph style should all be fully attributed. Character styles can't be fullly attributed, that is, they won't always be completely flattened.
- ProcessArchiveSubDocuments: Process all archive sub-objects and put the output in the main Flexiondoc output
- ProcessAttachmentSubDocuments: Process all attachments and put the output in the main Flexiondoc output
- ProcessEmbeddingSubDocuments: Process all embeddings and put the output in the main Flexiondoc output
- RemoveFontGroups: Replace font groups with references to individual fonts.
- IncludeTextOffsets: Include text_offset attribute on tx.p and tx.r elements.
- SeparateStyleTables: Enabling this flag will cause the style_tables subtree to be streamed to a separate output unit. This item is deprecated.
- UseFullFilePaths: Locators for externalized embeddings will contain full, absolute path names.
- BitmapAsBitmap: dr.image objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.image.
- ChartAsBitmap: ch.chart objects are converted to a graphic file and the resulting file is referenced by the locator child of the ch.chart.
- PresentationAsBitmap: pr.slide objects are converted to a graphic file and the resulting file is referenced by the locator child of the pr.slide.
- VectorAsBitmap: dr.drawing objects are converted to a graphic file and the resulting file is referenced by the locator child of the dr.drawing.
- GenerateSystemMetaData: When this flag is set, system metadata will be generated. This information is gathered through system calls and may adversely affect performance.
- NoBitmapElements: Bitmap graphics are suppressed; no dr.image content will appear in the converted document.
- NoChartElements: Charts are suppressed; no ch.chart content will appear in the converted document.
- NoPresentationElements: Presentation slides are suppressed; no pr.slide content will appear in the converted document.
- NoVectorElements: Vector drawings are suppressed; no dr.drawing content will appear in the converted document.
- These next four flags are mutually exclusive:
  - DefaultCharacterMapping: Default behavior: All text is mapped to Unicode, in tx.text elements.
  - NoCharacterMapping: All text is left in the original character set, in tx.utext elements.

- MapText: Text is mapped to Unicode where possible, unmappable text is left in the original character set.
- MappedAndUnmappedCharacters: Both mapped and unmapped text is included as an alt element containing tx.text and tx.utext.

Default

RemoveFontGroups

14.4 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.

Namespace

OutsideIn

Properties

- 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 that 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 sub documents 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 sub-document was not converted.

- TypeThreeFont: A type 3 font was encountered.
- UnsupportedShading: An unsupported shading pattern was encountered.
- InvalidHTML: An HTML parse error, as defined by the W3C, was encountered.

14.5 FileFormat Class

This class defines the identifiers for file formats.

Namespace

OutsideIn

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.

14.6 ObjectInfo Class

ObjectInfo provides all the information available about an Outside In Object (object may be an embedded object, a linked object, or a compressed file). This is a read only class where the technology fills in all the values.

Namespace

OutsideIn.Options

Properties

- Compression (Int32) The type of compression used to store the object, if known.
- Flags (ObjectInfoFlags) Flags indicating attributes of the object.
- FormatId (FileFormat) The format Identifier of the object.
- Name (String) Name of the object.

ObjectInfoFlags Enumeration

Bit fields to describe information about an object.

- PartialFile: Object would not normally exist outside the source document

- ProtectedFile: Object is encrypted or password protected
- UnsupportedCompression: Object uses an unsupported compression mechanism
- DRMFile: Object uses Digital Rights Management protection
- UnidentifiedFile: Object is extracted, but can not successfully identified
- LinkToFile: Object links to file, it can not be extracted
- EncryptedFile: Object is encrypted and can be decrypted with the known password

14.7 Option Interface

The Option Interface provides the methods and properties to retrieve information about an Outside In Option.

Package

com.oracle.outsidein.options

Accessors

- String getName() — Gets the name of the option
- String getDescription() — Gets the description of the option
- Class<?> getDataType() — Gets the type of the option value.
- Class<?>[] getItemTypes() — Gets the type parameters for option values that are generics
- EnumSet<Option.OutsideInProducts> getSupportingProducts() — Gets the list of products that support this option

Method

```
void set(OptionsCache exporter, Object objValue) throws OutsideInException;
```

This method sets the option to the exporter object and returns the exporter object itself.

- exporter — The exporter option
- objValue — Value of the option

Note:

If the type of objValue cannot be converted to the data type the option is expecting, an OutsideInException is thrown.

OutsideInProducts Enumeration

- HTMLEXPORT — Outside In HTML Export
- IMAGEEXPORT — Outside In Image Export
- PDFEXPORT — Outside In PDF Export
- SEARCHEXPORT — Outside In Search Export

- WEBVIEWEXPORT — Outside In Web View Export
- XMLEXPORT — Outside In XML Export

14.8 OutsideIn Class

This is a utility class that creates an instance of an Exporter object on request.

Namespace

OutsideIn

Methods

```
static Exporter NewLocalExporter()
```

This method creates an instance of an Exporter object. It returns a newly created Exporter object.

```
static Exporter NewLocalExporter(Exporter source)
```

This method creates and returns an instance of an Exporter object based on the source Exporter. All the options of source are copied to the new Exporter. The source and destination file information will not be copied.

14.9 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.

Namespace

OutsideIn

14.9.1 OutsideInCastException Class

This exception is thrown when an invalid value is provided as an option value.

This class derives from the OutsideInException class. This class has no public methods or properties except those of the parent Exception class.

Namespace

OutsideIn

14.10 XMLReference Class

The XMLReference class is a data class used to define the XML definition reference to be used.

Namespace

OutsideIn.Options

Constructors

`XMLReference()`

Create an instance of a XMLReference object using No XML defintion reference

`XMLReference(XMLReference.ReferenceMethodValue, String)`

Create an instance of a XMLReference object to provide a DTD/XSD

ReferenceMethodValue Enumeration

This enumeration is used to set whether Export will reference a schema, a DTD, or no reference when generating output.

- DTD: Document Type Definition (DTD)
- XSD: Extensible Schema Definition
- NONE: No definition reference

Index