# Oracle® Outside In Transformation Server

Developer's Guide

Release 8.5.2

**E12868-09**

April 2015

ORACLE

Oracle Outside In Transformation Server Developer's Guide, Release 8.5.2

E12868-09



Primary Author: Mike Manier

# Contents

## A SOAP Data Types and Options

# Preface

This document describes the installation and usage of the Outside In Transformation Server.

## Audience

This document is intended for developers who are investigating the Outside In Transformation Server as a way of running Outside In SDKs.

## Documentation Accessibility

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

### Access to Oracle Support

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

## Related Documents

For more information, go to:

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

and click on Outside In Technology.

## Conventions

The following text conventions are used in this document:

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

# 1

# What Is Transformation Server?

This chapter is an introduction to Transformation Server. Transformation Server is an add-on component to the Outside In Export SDKs. It provides an alternative means of controlling Outside In, by supplying a runtime environment that manages file-export operations in processes that execute independently of your application. In other words, it moves the execution of Outside In from an in-process component model to a client-server model.

Major features of Transformation Server include:

- Service-oriented architecture that allows your application to control export operations with complete isolation from the memory and execution space of the export process, for maximum fault tolerance.
- Multiple interfaces - Transformation Server includes interfaces in C/C++, Java, or the SOAP protocol.
- Process management - Transformation Server can monitor its export processes and will restart them in the event of an exception or an infinite loop.
- Support for all of the Outside In Export SDKs.
- A published add-on interface that allows the developer to implement custom input/output.
- A published add-on interface that allows customer or third- party export code to be integrated into Transformation Server.

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

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

and click on Outside In Technology.

This chapter includes the following sections:

# What's New in this Release

- The updated list of supported formats is linked from the page http://www.outsideinsdk.com/. Look for the data sheet with the latest supported formats.
- Support has been added for Oracle Linux x86-64 R7.

# Components of Transformation Server

Transformation Server consists of several components that interact with each other and your application. Some of these components are optional, depending on the way you wish to configure your application.

## The Transformation Agent (TSAGENT)

This is an executable that hosts the Outside In Export SDKs. It is controlled through a SOAP version of the Outside In Export APIs. This executable can be used directly by an application or an application server, or accessed indirectly through the Transformation Manager.

## The Transformation Manager (TSMANAGER)

The Transformation Manager is an executable that is used as a central controller for Transformation Agents. The Transformation Manager controls a pool of Transformation agents, each of which is used to perform Export operations. The Transformation Manager presents the same SOAP version of the Outside In API as does the Transformation Agent, and will manage a queue of transformation requests. The Transformation Manager also monitors the status of Transformation Agents, and will kill and restart an agent if it has become unresponsive. If your application already has app server functionality, you may decide not to use the Transformation Manager.

## The C Client Module (SCCTS)

This module allows an application written in C to control Transformation Server through a C API that resembles the embedded version of the Outside In C API as closely as possible. This module handles all SOAP communication between an application and Transformation Server, and is the fastest way to migrate an existing application from using the embedded version of the Outside In SDK to Transformation Server.

## The Java Client (TSAPI)

The Java client is a set of JAR files that provide objects that represent a Java version of the Outside In SDK API, and handle all SOAP communication between a Java application and Transformation Server. This client allows a Java application to use Transformation Server without the need to write any C/C++ code.

# Architecture

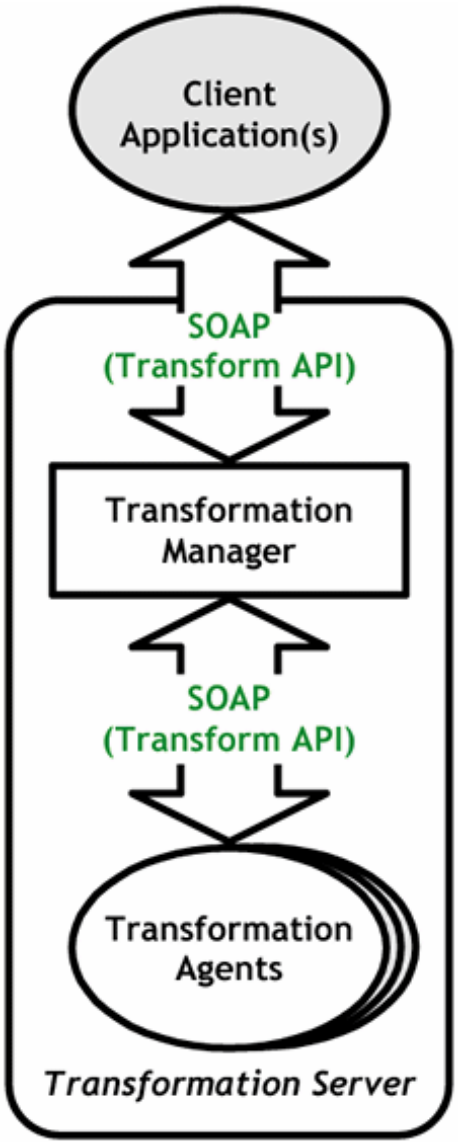
This section describes the system software architecture.

## The Transformation Manager

Client applications control document transformations by communicating with an application called the **Transformation Manager**, which itself manages a pool of processes called Transformation Agents (see The Transformation Agent).

The following ilustration depicts this communication.

***Figure 1 Client Application Communication***




Requests to transform documents are distributed among the available running Transformation Agents. If all available Transformation Agents are in use, transformation requests are queued until an agent is available.

The interface that controls transformations is called the Transform API. The Transform API provides the means to initiate a transformation and pass parameters to the technology that will perform the transformation.

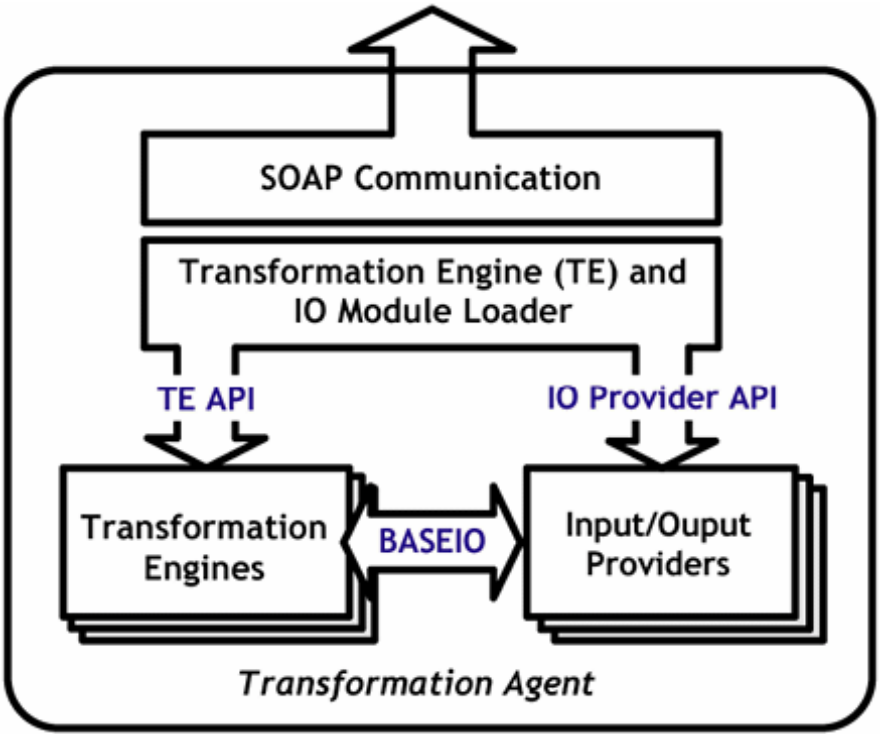
The Transform API is accessed via SOAP messages, through C-language function calls using the C Client Module, and /or through Java methods presented by the Java Client object.

## The Transformation Agent

The Transformation Agent (TSAGENT) is the worker process in which a transformation actually occurs. Transformation Agents host the available Transformation Engines and Input/Output Providers.

The following illustration depicts the transformation agent.

***Figure 2 Transformation Agent***




Transformation Engines and IO Providers are loadable modules (DLLs) that execute within the Transformation Agent process.

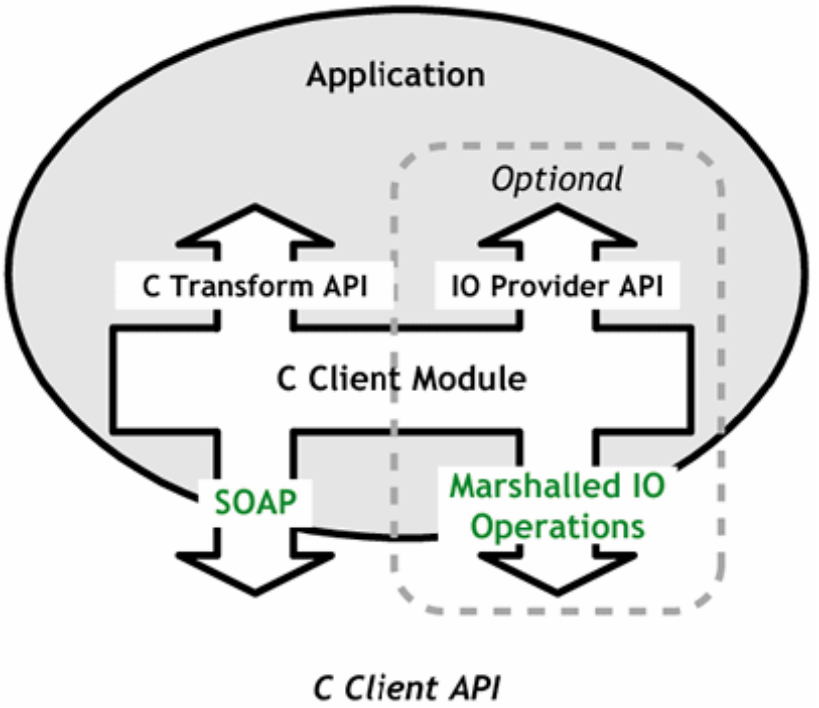
The C language interfaces between the Transformation Agent, the Transformation Engines, and IO Providers are fully documented in this guide.

Combined with the extensibility of the Transform API, this allows third party repositories and transformation technologies to be incorporated into Transformation Server without any changes to the infrastructure itself.

## C Language Client Module (sccts)

Transformation Server's SCCTS client module runs inside an application's process, and presents a C language interface through which the application can invoke all of the functionality of Transformation Server. SCCTS will provide all of the SOAP/HTTP communication, and will support the same IO Provider API that is available on the server.

*Figure 3 IO Provider API*



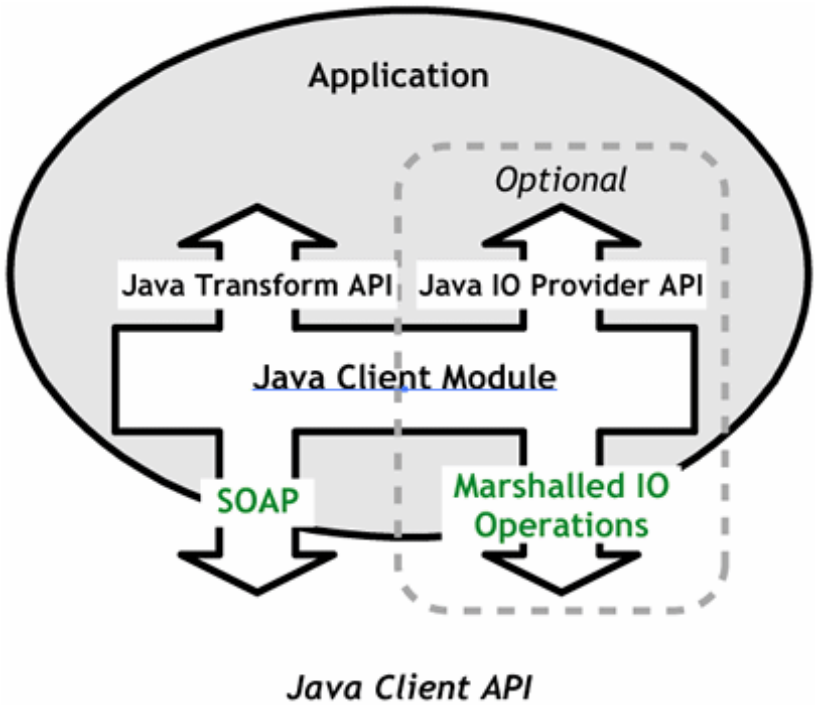

If an application provides custom IO on the client side, the interface module will marshal IO operations between the client process and Transformation Agent, over a socket connection.

For a developer, the only difference between implementing an IO provider on the client versus on the server is that client-side IO provider code is not required to exist in a shared library or DLL, and is invoked through the Transform API rather than a configuration file.

## Java Client Object

Much like the C/C++ client module, the Java client object provides a Java version of the Transform API (see Initiating Transformations Using the Java API) that allows an application to access Transformation Server directly through code, rather than through the SOAP protocol.

*Figure 4 Java Client API*




The Java client object will handle all of the SOAP communication with Transformation Server, and will return the results back to the calling application.

If the client application developers wish to provide their own Java-based input/output, they may do so by implementing the Java version of the IO Provider API. (Note: the IO Provider API on the server side is C-based only.)

## 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, ts/sdk/ts_win-x86-32_sdk). Under each of these directories are the following three 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 named '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, ts/sdk/ts_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.

# Copyright Information

The following notice must be included in the documentation, help system, or About box of any software that uses any of Oracle's executable code:

**Outside In Image Export, PDF Export, Search Export, XML Export, HTML Export © 1991, 2015 Oracle**.

The following notice must be included in the documentation of any software that uses Oracle's TIF6 filter (this filter reads TIFF and JPEG formats):

**The software is based in part on the work of the Independent JPEG Group**.

# 2

# Installing and Running Transformation Server

This chapter details the fundamental steps you must take to run Transformation Server and begin generating output using the tsmanager and/or tsagent modules.

This chapter includes the following sections:



## Installation

To install Transformation Server, download the archive from the Oracle site (`http://edelivery.oracle.com/`). Place the download into a directory that contains a previously downloaded Outside In installation on your local drive to complete installation.

---

**Note:**

At least one of the Outside In Export products needs to be installed before Transformation Server is downloaded.

---

---

**Note:**

Transformation Server, its configuration files, and the Outside In Export technologies are designed to reside in the same directory. Placing them elsewhere may result in errors.

You will need to set the TSROOT variable to the location of the Transformation Server installed SDK. For example, for a Linux version of Transformation Server, you would set TSROOT=/user/jsmith/ts/ts_linux-x86-32_sdk/sdk.

---

### 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.

## Motif Library Compatibility Information

On some Linux installations, particularly newer ones, the Motif libraries that are installed are not compatible with the libraries that are used to build the Outside In technology. This is known to be the case with most of the SuSE installations, for example. It is likely that you have a binary incompatibility if you try to build one of the Xwindows-based sample applications included with this product and see an error at compile time that looks like the following

```
warning: libXm.so.1, needed by ../../libsc_vw.so, may conflict with libXm.so.3
```

Problems can also be seen when trying to convert graphics files. For example, zero byte graphics files will be generated if our 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 will print a list of the dependencies that this library has. If the line for the Motif library looks like the following:

```
libXm.so.1 => not found
```

then your system may not have a compatible Motif library installed.

The proper solution to both of these problems is to install a compatible Motif library and use it to build your application. Often, the installation discs for your particular Linux platform will have the proper libraries. If your installation discs do not have the libraries, instructions for downloading a binary rpm can be found at http://www.lesstif.org/download.html. Remember that if you are doing development, you will also need the proper header files, as well.

The Motif library versions used by Oracle when building and testing the Outside In binaries are:

- x86 Linux: OpenMotif v. 2.2.3

## Visual C++ Redistributable Dependency

This product requires the Visual C++ libraries included in the Visual C++ Redistributable Package available from Microsoft. Transformation Server requires the x86 Windows package. Users can dowload the required library from http://www.microsoft.com/downloads/details.aspx?FamilyId=90548130-4468-4BBC-9673-D6ACABD5D13B&displaylang=en.

To deploy Visual C++ libraries using the Visual C++ Redistributable Package, perform the following steps:

1. Copy the Visual C++ Redistributable Package (vcredist_x86.exe) to the target computer.
2. Run vcredist_x86.exe on the target computer. This installs all Visual C++ libraries as shared assemblies. On a target computer with support for manifest-based binding of applications to their dependencies (Windows XP Home Edition,

Windows XP Professional, Windows Server 2003, Windows Vista), the libraries are installed in the WinSxS folder. On a computer without such support (Windows 2000), the libraries are installed to both the WinSxS and System32 folders.

3. Your application is ready to be run.

## Environment Variables on UNIX

Transformation Server does not require the setting of the environment variable LD_LIBRARY_PATH in order to run. However, it does require the setting of the PATH environment variable. Transformation Server's process manager (tsmanager) launches child processes, and in order to be able to locate those processes it needs the PATH environment variable to include the path to the directory where Transformation Server is installed.

Additional environment variables for UNIX may need to be set depending upon the Outside In export technology you are invoking via Transformation Server. Please consult the manual for the export technology you are using as to whether it needs additional environment variables to be set for your specific UNIX platform.

## Libraries and Structure

Here is an overview of the files contained in the main installation directory for this product:

| File | Description |
|---|---|
| ACE-5.8.1 | Third party ACE library that provides process control and thread synchronization |
| sccts | C API library for Transformation Server Clients |
| ts_buffered_io | Transformation Server buffered IO library |
| ts_components | Transformation Server support library |
| ts_engine_options | Transformation Server generic options handling library |
| ts_file_io_module | Transformation Server file IO library |
| ts_hx_engine | HTML Export engine for use by tsagent |
| ts_ix_engine | Image Export engine for use by tsagent |
| ts_logging_facility | Transformation Server logging support library |
| ts_msg_logger_app | Transformation Server logging process |
| ts_pdf_engine | PDF Export engine for use by tsagent |
| ts_riot_iop_module | Transformation Server Client side, redirected IO library |
| ts_riotstub | C API support library for Transformation Server Clients |
| ts_soap_ext | Transformation Server SOAP support library |
| ts_soap_std | Transformation Server SOAP support library |
| ts_soap_ta_client | Transformation Server SOAP support library |
| ts_soap_ts_client | Transformation Server SOAP support library |
| ts_soap_ts_server | Transformation Server SOAP support library |
| ts_soap_tss | Transformation Server SOAP support library |

| File | Description |
|---|---|
| ts_sx_engine | SearchML Export engine for use by tsagent |
| ts_url_iop_module | Transformation Server HTTP IO library |
| ts_utils | Transformation Server support library |
| ts_xx_engine | XML Export engine for use by tsagent |
| tsagent | Transformation Server agent process |
| tsmanager | Transformation Server manager process |
| tsapi.jar | Java API library for Transformation Sever Clients |
| tools.jar | Java classes used by configserver and option_set_editor utilities |
| configserver | Java GUI application for editing the server_startup.xml file |
| option_set_editor | Java GUI application for editing the agent_option_sets.xml file |

# Running Transformation Server

Transformation Server's main application is tsmanager. When this program is started, it will in turn start up Transformation Agents (tsagent) and initiate communication with them. tsmanager will then listen for incoming transformation requests. Incoming requests for transformations are distributed among the running Transformation Agents, who return the results of the requests to tsmanager, which will then send those results back to the original requestor.

tsagent can be started in standalone mode apart from tsmanager as well, for applications where reduced resource usage is desirable.

This section deals with these applications and their command-line parameters.

**Note:**

If any of the following .xml files that tsmanager and its agents depend upon change while tsmanager is running, then it must be shut down and restarted.

- server_startup.xml
- agent_option_sets.xml
- agent_engine_list.xml
- agent_iospect_types.xml

## tsmanager

The following is an overview of product details.

### Startup Parameters

The user can alter tsmanager's parameters via the command line, the XML configuration file named server_startup.xml, or both. For parameters that can be set both ways, the command line overrides server_startup.xml. For both methods,

parameter string values are case-sensitive. If server_startup.xml is malformed or missing an option, tsmanager will resort to defaults if it can avoid a bailout.

## Command Line Options with Parameters

The following command line options have parameters. The command --help provides a list of these as well as their shorthands (e.g., --host = -s).

#### --host

The hostname or IP address to use when listening to transformation requests. Value may be a name, such as "server.host.com" or an IP address such as "127.0.0.1;" the host address must be valid for the machine on which tsmanager is running. If no host name is specified on the command line or in the configuration file, tsmanager will listen to incoming requests on all available addresses. If no host is specified on either the command line or in server_startup.xml, tsmanager will listen for requests on all of the computer's available IP addresses. The server_startup.xml parameter is the <serverName> subelement of <ConnectionsInfo>.

#### --port

The TCP port number on which tsmanager will listen for incoming requests. There is no default value; this parameter must be specified. The server_startup.xml parameter is the <port> subelement of <ConnectionsInfo>.

#### --pipedir

This parameter is only valid on UNIX systems.

This is the directory where pipes will be created. The location must be local to the machine. The default value is /tmp. The server_startup.xml parameter is the <pipeDir> subelement of <logInfo>.

#### --numagents

The number of simultaneously running Transformation Agents that will be available to handle requests for documentation transformations. The default value is 4. The server_startup.xml parameter is the <poolSize> subelement of <agentsInfo>.

## Command Line Flags

The following command line flags are used.

#### --trace_on

If set, this option prints supplemental diagnostic information to the log file.

#### --version

This option returns the version number of the application and copyright information.

## Command Line Syntax

The tsmanager command line syntax is simple:

```
tsmanager --parameter1 value1 --parameter2 value2 ...
```

Both the long option and short option can be used for syntax, for example, the port command long option is "--port" and the short option is "-p". Use the --help (-h) command to show the short options. Parameters and their values are separated by one or more spaces or tabs.

The following parameters are supported:

- cfgfile: The parameter cfgfile causes tsmanager to update the parameter values in server_startup.xml with the other parameter values specified on the same command line, then exit immediately without initiating any transformation operations.
- host: Described in Command Line Options with Parameters.
- loghostname: Logs the client host name (slower than logip).
- logip: Logs the client's IP address.
- numagents: Described in Command Line Options with Parameters.
- pipedir: Described in Command Line Options with Parameters.
- port: Described in Command Line Options with Parameters.
- trace_on: Described in Command Line Flags.
- version: Described in Command Line Flags

### Command Line Examples

The following command line will cause tsmanager to listen for incoming transformation requests on TCP port 90 of IP address 127.0.0.1 (localhost).

```
tsmanager --host 127.0.0.1 --port 90
```

The following command line will cause tsmanager to listen for incoming transformation requests on TCP port 100 and use a pool of 3 transformation agents to handle transformation requests.

```
tsmanager --port 100 --numagents 3
```

### Logging

When you launch tsmanager, it spawns a logging application called ts_msg_logger_app. This application logs server activity based on parameters specified in the server_startup.xml configuration file (the configuration file is discussed in the following section). The log can be rotated based on a maximum file size (the rotateSize parameter in the logInfo section of the server_startup.xml file) or by time of day (the rotateTime parameter in the logInfo section of the server_startup.xml file). When the log is rotated, the old log file is not deleted.

### Configuration File

The file server_startup.xml resides in the same directory as tsmanager. If this file is not present, you must specify at least the port number on the tsmanager command line.

If the rotate_time value of the logInfo section of the configuration file is filled with a valid, non-empty time, rotate_size can be zero or empty. However, if the rotate_time value is empty and the rotate_size value is zero, ts_msg_logger_app will not start and will print out an error message to the console. Also, if both rotate_time and rotate_size have non-empty values, tsmanager will always use the rotate_time value and ignore the rotate_size value.

The path value must be set or ts_msg_logger_app will not start.

### Configuration File Example

```
<TsServerStartup xsi:type="tss:TsServerStartup"
... other attributes deleted for clarity... >
   <agentsInfo xsi:type="ts:agentsInfo">
      <poolSize xsi:type="xsd:unsignedInt">4</poolSize>
   </agentsInfo>
   <connectionsInfo xsi:type="tss:connectionsInfo">
      <serverName xsi:type="xsd:string"></serverName>
      <port xsi:type="xsd:unsignedInt">60611</port>
   </connectionsInfo>
   <logInfo>
      <host>128.26.53.89</host>
      <port>90</port>
      <path>c:\logs</path>
      <rotate_time>23:00</rotate_time> <!-- HR:MN -->
      <rotate_size>5</rotate_size> <!-- in MB -->
   </logInfo>
</TsServerStartup>
```

## tsagent

In addition to its typical usage, running in tandem with tsmanager, tsagent can be started in standalone mode (using the standalone flag), allowing low-overhead, single-threaded communications with the server. Here is a guide to tsagent's parameters.

### Command Line Flags with Parameters

The following are command line flags with parameters.

#### --host

The hostname or IP address to use when listening to transformation requests. Value may be a name, such as "server.host.com" or an IP address such as "127.0.0.1;" the host address must be valid for the machine on which TSagent is running. If no host name is specified on the command line or in the configuration file, TSagent will listen to incoming requests on all available addresses. There is no default value; this parameter must be specified.

#### --port

The TCP port number on which tsmanager will listen for incoming requests. There is no default value; this parameter must be specified.

### Command Line Flags

The following are other command line flags.

#### --help

This option presents a quick summary of the command line options for tsagent.

#### --oneclient

This flag will allow tsagent to connect to one client and when that client ends the connection, tsagent will exit.

#### --standalone

This flag is required if you wish to run tsagent in standalone mode separately from tsmanager.

**--stdinout**

This flag allows the standalone agent to get input from stdin and write output to stdout.

**--version**

This option returns the version number of the application and copyright information.

# Configuration Files

There are several XML configuration files used by Transformation Server to store its configuration information. These files are located in the same directory as the Outside In binaries. A developer may add IO Providers or Transformation Engines to the Transformation Server core by modifying these files.

- agent_iospec_types.xml: Contains the list of input/output providers installed on the server. Each element in the list maps an IO "specification type" (for example, path or url) to a module that contains the code that implements the IO Provider interfaces for this type.
- agent_engine_list.xml: Contains the list of Transformation Engines available on the server.
- agent_option_sets.xml: Contains the predefined sets of transformation options.
- server_startup.xml: Contains startup parameters that affect the tsmanager application. If this file is not present, tsmanager parameters must be specified on its command line.

## Examples

The following are examples from provided files.

### agent_iospec_types.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?file version="1.0"?>
<IoSpecTypeToModuleMap xsi:type="tss:IoSpecTypesList"
   other attributes deleted for clarity...>

   <SpecType xsi:type="tss:SpecType">
      <!-- For local file system and shared file system paths-->
      <Name xsi:type="xsd:string">path</Name>
      <Module xsi:type="xsd:string">ts_file_io_module</Module>
   </SpecType>
   <SpecType>
      <!--  used internally for client-side redirected IO
         support -->
      <Name xsi:type="xsd:string">riot</Name>
      <Module xsi:type="xsd:string">ts_riot_iop_module</Module>
   </SpecType>
   <SpecType>
      <!-- For files that can be read (GET) and written (PUT)
               via HTTP-->
      <Name xsi:type="xsd:string">url</Name>
      <Module xsi:type="xsd:string">ts_url_iop_module</Module>
   </SpecType>
</IoSpecTypeToModuleMap>
```

### agent_engine_list.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?file version="1.0"?>
<EngineList xsi:type="tss:EngineList"
      other attributes deleted for clarity...>
<Engine xsi:type="tss:Engine">
<EngineName xsi:type="xsd:string">HTML Export Engine</EngineName>
<EngineModule xsi:type="xsd:string">ts_hx_engine</EngineModule>
<OutputFormatNames xsi:type="tss:StringList">
<Name xsi:type="xsd:string">html</Name>
<Name xsi:type="xsd:string">mhtml</Name>
</OutputFormatNames>
</Engine>
<Engine xsi:type="tss:Engine">
<EngineName xsi:type="xsd:string">Image Export Engine</EngineName>
<EngineModule xsi:type="xsd:string">ts_ix_engine</EngineModule>
<OutputFormatNames xsi:type="tss:StringList">
<Name xsi:type="xsd:string">bmp</Name>
<Name xsi:type="xsd:string">gif</Name>
<Name xsi:type="xsd:string">jpeg</Name>
<Name xsi:type="xsd:string">png</Name>
<Name xsi:type="xsd:string">tiff</Name>
</OutputFormatNames>
</Engine>
<Engine xsi:type="tss:Engine">
  <EngineName xsi:type="xsd:string">Search Export Engine</EngineName>
  <EngineModule xsi:type="xsd:string">ts_sx_engine</EngineModule>
  <OutputFormatNames xsi:type="tss:StringList">
  <Name xsi:type="xsd:string">page-ml</Name>
  <Name xsi:type="xsd:string">search-ml</Name>
  <Name xsi:type="xsd:string">search-ml-2</Name>
  <Name xsi:type="xsd:string">search-ml-3</Name>
  <Name xsi:type="xsd:string">search-ml-3-1</Name>
  <Name xsi:type="xsd:string">search-ml-3-2</Name>
  <Name xsi:type="xsd:string">search-html</Name>
  <Name xsi:type="xsd:string">search-text</Name>
  </OutputFormatNames>
  </Engine>
<Engine xsi:type="tss:Engine">
  <EngineName xsi:type="xsd:string">XML Export Engine</EngineName>
  <EngineModule xsi:type="xsd:string">ts_xx_engine</EngineModule>
  <OutputFormatNames xsi:type="tss:StringList">
  <Name xsi:type="xsd:string">flexiondoc-4</Name>
  <Name xsi:type="xsd:string">flexiondoc-5</Name>
  <Name xsi:type="xsd:string">flexiondoc_5</Name>
  <Name xsi:type="xsd:string">flexiondoc-5-1</Name>
  <Name xsi:type="xsd:string">flexiondoc-5-2</Name>
  </OutputFormatNames>
  </Engine>
<Engine xsi:type="tss:Engine">
  <EngineName xsi:type="xsd:string">PDF Export Engine</EngineName>
  <EngineModule xsi:type="xsd:string">ts_pdf_engine</EngineModule>
  <OutputFormatNames xsi:type="tss:StringList">
  <Name xsi:type="xsd:string">pdf</Name>
  </OutputFormatNames>
  </Engine>
  </EngineList>
```

### agent_option_sets.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?file version="1.0"?>
<OptionSets xsi:type="tss:OptionSetList"
      other attributes deleted for clarity...>

   <OptionSet xsi:type="tss:OptionSet">
      <Name xsi:type="xsd:string">Netscape 6.2</Name>
```

```
      <Options xsi:type="tss:OptionList">
         <Option xsi:type="ts:Option">
            <name xsi:type="xsd:string">flavor</name>
            <value xsi:type="ts:FlavorEnum">netscape4.0</value>
         </Option>
         <Option xsi:type="ts:Option">
            <name xsi:type="xsd:string">defaultFont</name>
            <value xsi:type="tsDefaultFont">
               <fontname xsi:type="xsd:string">Times New Roman
                 </fontname>
               <height xsi:type="xsd:unsignedShort">9</height>
            </value>
         </Option>
      </Options>
   </OptionSet>
   <OptionSet xsi:type="tss:OptionSet">
      <-- ... more option sets... -->
   </OptionSet>
</OptionSets>
```

**server_startup.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<?file version="1.0"?>
<TsServerStartup xsi:type="tss:TsServerStartup"
      other attributes deleted for clarity...>
   <agentsInfo xsi:type="ts:agentsInfo">
      <poolSize xsi:type="xsd:unsignedInt">4</poolSize>
   </agentsInfo>
   <connectionsInfo xsi:type="tss:connectionsInfo">
   <serverName xsi:type="xsd:string">127.0.0.1</serverName>
   <port xsi:type="xsd:unsignedInt">9999</port>
</connectionsInfo>
</TsServerStartup>
```

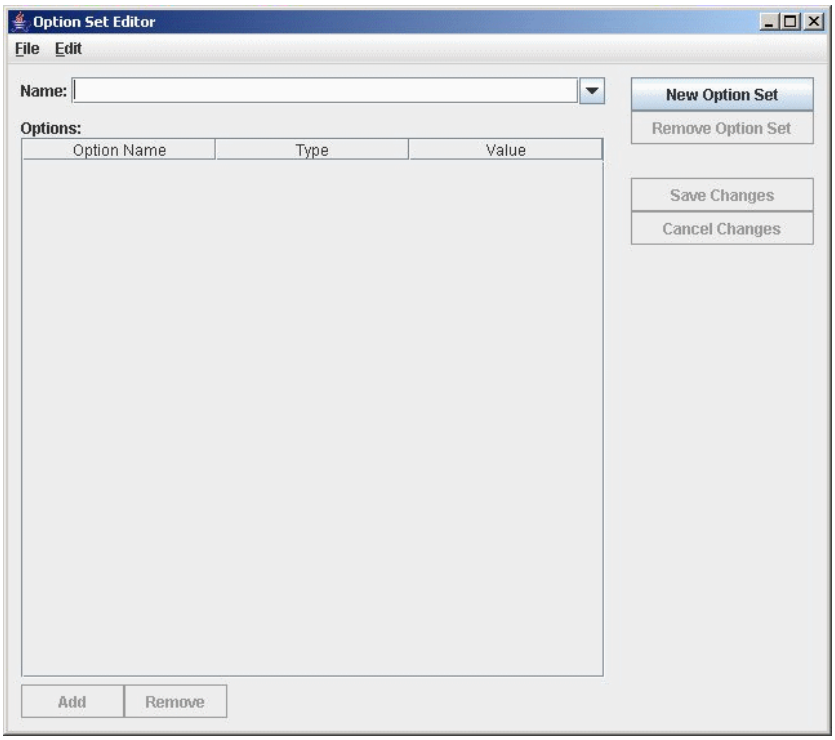
# The Option Set Editor

To facilitate easy creation of option sets for transformations, an Option Set Editor application is included with Transformation Server. This application is launched using a batch file called option_set_editor, located in the installation's root directory.

## Using the Option Set Editor

Once launched, the editor displays its main screen, shown here:

***Figure 1 Option Set Editor Main Screen***




Here are the steps to follow to create an option set using this application.

1. Click **New Option Set** to create a new set.
2. Type a name for the new set in the Name field.
3. Add an option by either clicking the **Add** button at the bottom of the main screen, or clicking **Edit** then **Add Option**.
4. By default, the new option will be xsd:boolean, but this value can be changed by double-clicking on **xsd:boolean** in the Type column, and then selecting a new option from the drop-down list that appears. Similarly, you can change the value for any option in the set by double-clicking on the current value in the Value column, and then either selecting a value from the drop-down menu (if applicable) or typing a value into the field.
5. If you wish to remove an option that you've added, click it to highlight it, and then click the **Remove** button, or click **Edit** then **Remove Option**. You can also remove all options from the current set by clicking **Edit** then **Remove All Options**.
6. To begin a new option set, you can click **New Option Set** or **Edit** then **New Option Set**.
7. When you are ready to save a set to an XML file, click **Edit** then **Save** or **Edit** then **Save As**.
8. You can open an existing option set for editing by clicking **File** then **Open**. If at any point you wish to revert to the file's initial state prior to editing, click the **Cancel Changes** button. Clicking this button when working with a set that has not yet been saved simply brings you back to the blank default start screen.

**Note:**

The Option Set Editor can make intelligent decisions about when to base64-encode text. For example, if a user creates a stringData entry in the Option Set Editor and the character set of the string is not UTF-8-encoded, the editor will automatically base64-encode the string and set the base64 flag to true. Such behavior explains the possible presence of <base64> tags in the output.

## Extending the Functionality of Transformation Server

Transformation Server exposes several internal APIs to allow your application to extend the core functionality to better integrate with your application.

- The Transformation Engine specification allows your application to integrate non-Outside In export software into the Transformation Server framework. See Transformation Engine Specification for details.
- The IO Provider specification allows your application to extend the input and output functionality beyond the file system. See IO Provider Specification for details.

# 3

# Initiating Transformations Using the SOAP API

This chapter discribes the use of the SOAP API. All SOAP requests sent to, and all SOAP responses sent by Transformation Server are UTF-8 encoded UNICODE. To use Transformation Server in a SOAP application, the application should send the TransformRequest message defined in the transform.wsdl.

Three other .wsdl files are included with Transformation Server:

- **transform_net.wsdl** should be used by those working with Microsoft's DevStudio .NET development tools.
- **transform_net_2005.wdsl** should be used by those working with Visual Studio 2005 or Visual Studio 2008.
- **transform_axis.wsdl** should be used by those working with Apache.org's Axis development tools.

Microsoft's tool sproxy does not support the declaration of SOAP arrays, which are part of the Transformation Server SOAP interface. As a result, none of the .wsdl files provided with Transformation Server can be successfully interpreted by sproxy.

This chapter includes the following sections:



## TransformRequest

Initiates a transformation based on criteria contained within the message.

### Prototype

```
<message name="TransformRequest">
   <part name="source" type="ts:IOSpec"/>
   <part name="sink" type="ts:IOSpec"/>
   <part name="outputFormat" type="xsd:string"/>
   <part name="optionSet" type="xsd:string"/>
   <part name="options" type="ts:ArrayOfOption"/>
</message>
```

- source: A ts:IOSpec element defining the transformation source.
- sink: A ts:IOSpec element defining the transformation sink.
- outputFormat: Output format name.

- optionSet: Option set name.
- options: A temporary option set whose options will overwrite the corresponding options of the persistent option set specified above (optionSet).

# TransformationResponse

Each TransformRequest message is met with a TransformResponse message detailing the success or failure of the transformation.

### Prototype

```
<message name="TransformResponse">
   <part name="result" type="xsd:unsignedInt"/>
   <part name="resultMsg" type="ts:stringData"/>
   <part name="resultDocs" type="ts:ArrayOfIOSpec"/>
</message>
```

- result: Numeric error code. 0 means the transformation was successful.
- resultMsg: Text error message. This may be empty, depending on the error code.
- resultDocs: An array of ts:IOspec, one for each output document created. This may be empty if an error occurred.

# Transformation Server's HTTP GET/POST Interface

The full functionality of Outside In Transformation Server is available through the SOAP/XML Web Services interface. In addition to this interface, the server also provides a direct interface via URL-encoded HTTP GET or POST requests.

The HTTP GET and POST interfaces provide a fast means to integrate Outside In Transformation Server into an existing application or web page. They have the added benefit of providing access to transformations without writing a line of code. All that's required is creating a properly encoded HTML link or a simple HTML form.

## Differences Between the HTTP POST/GET and Full SOAP/XML Interfaces

The GET and POST interfaces have the following limitations:

- Transformation options specific to the output format may only be specified via a named server-side option set. (No specific individual options may be set.)
- The transformation parameters must be provided in UTF-8 strings (prior to being URL-encoded).

## Using the GET/POST Interface

The HTTP interface accepts the following parameters. These parameters must be specified in URL-encoded format. For the HTTP GET interface, they are provided as query parameters appended to the Transformation Server URL. For the HTTP POST interface, they are contained in the body of the POST request.

In either case, the response to the HTTP request is identical to the one returned from the SOAP interface. It is an HTTP response the body of which contains a SOAP response encoded in XML : a TransformResponse inside a SOAP Envelope and Body.

URL encoding means that all characters that are reserved or forbidden in a URL must be represented by an escape sequence consisting of a percent sign and the hex

representation of their value (for example, the ":" character is "%3A", "\" is "%5C", the space character is "%20", etc.).

- source: This is the input file for the transformation. This string is required to be UTF-8 encoded UNICODE, prior to being URL-encoded for transmission. (Required)
- sink: This is the output file for the transformation. This string is required to be UTF-8 encoded UNICODE, prior to being URL-encoded for transmission. (Required)
- sourcetype: This describes the type of IO specification used for the source file. If not present, Transformation Server will inspect the source parameter and attempt to guess the specification type.
- sinktype: This describes the type of IO specification used for the sink file. If not present, Transformation Server will inspect the sink parameter and attempt to guess the specification type.
- format: Specifies the output format for the transformation. If this parameter is not present, HTML is assumed to be the desired output format. The valid output formats are contained in the configuration file called agent_engine_list.xml.
- optionset: Specifies the option set in agent_option_sets.xml to be used in the transformation. If the set specfied is not present in agent_option_sets.xml or this parameter is not set, no option set will be used.

Transformation Server's ability to guess the type of an IO specification is very limited. Transformation Server will assume that the specification is for a file system path unless the specification begins with "http://", in which case it will assume the specification is a URL.

## Example

The following are examples to demonstrate the interface.

### Using the GET Interface

To request a transformation with an HTTP GET request, the transformation parameters and their values must be appended as query parameters to the URL address used for Transformation Server transformation requests.

For example, if Transformation Server is running on port 9000 of the local host, the input document is c:\files\sample.doc and the output document is c:\output \sample.htm, the appropriate URL for the transformation is:

```
http://localhost:9000/transform?source=c%3A%5Cfiles%5Csample.
   doc&sink=c%3A%5Coutput%5Csample.htm
```

### Using the POST Interface

The POST interface uses the same parameters and URL encoding as the GET interface, but sends the parameters as the body of an HTTP POST instead of appending them to the URL.

An HTML form may be the easiest way to generate a POST request for a transformation. The following HTML could be used:

```
<FORM name="SOAPREQ" ACTION="http://localhost:9000/transform"
   METHOD="post" enctype="application/x-www-form-urlencoded">
<p>Input file: <br/>
```

```
<INPUT TYPE="TEXT" NAME="source" size="60"><br/>
</p>
<p>Output file: <br>
<INPUT TYPE="TEXT" NAME="sink" size="60"><br/>
</p>
<p><INPUT TYPE="SUBMIT" value="Run Transformation"></p>
</FORM>
```

### The HTTP Response

Transformation Server's response to an HTTP request for a transformation is a SOAP XML document describing the results of the transformation request. This response is identical to that returned by the SOAP HTTP interface. This response may be parsed and consumed directly by your application, or an XSL stylesheet may be used to present this information to an end user.

## Sample Pages

Use of the HTTP POST and GET interfaces to Transformation Server are demonstrated by the sample HTML pages TSGET.HTM and TSPOST.HTM. While the POST and GET interfaces may be used from any application that can send HTTP requests (not just a browser), inspecting the HTML source of these pages should give you a good idea of how the GET and POST interfaces may be used.

### tsget.htm

TSGET.HTM is a web page that uses JavaScript to generate a URL that will request a transformation from Transformation Server. The XML response to that transformation request will be displayed in a new browser window.

If your browser is Microsoft Internet Explorer 6, you have the option of using the accompanying XSL stylesheet TSRESP.XSL to format the XML response as a Web page. Otherwise, the results will be displayed in whatever way your browser chooses to display XML documents.

### tspost.htm

TSPOST.HTM is a page that contains a very simple HTML form. This form generates an HTTP POST that will initiate a transformation and return the XML response to the browser. The response will be displayed in whatever manner your browser displays XML files.

TSPOST.HTM must be edited prior to using it. The HTML form contains a URL that must be modified to reflect the TCP address where your installation of Transformation Server can be found.

# 4

# Initiating Transformations Using the C/C++ API

This chapter describes the use of the C/C++ API. To use Transformation Server in a C or C++ application, the application should load the sccts module and communicate with its API functions. General operation consists of the following steps:

- Initialize the sccts module by calling TSInit.
- Set transformation parameters by calling TSSetOption or TSSetOptionById.
- Perform transformation(s) by calling TSRunTransform.
- Before unloading the sccts module, call TSDeInit.

The module SCCTS implements the Transformation Server C/C++ interface. Source files that use SCCTS should include the header file sccts.h, and make sure that the other header files included in the Transformation Server SDK are in the project's include path. Projects using SCCTS should link with SCCTS.LIB, which is included in the Transformation Server SDK. Additionally, the modules that reside in the root level of the Outside In Transformation Suite directory are required to be in the same directory as SCCTS.

---

**Note:**

The return values listed for these functions are only a selection of the most common error messages returned. For each function, other error messages are possible. These messages can be found in sccerr.h and tserr.h.

---

This chapter includes the following sections:

# TSInit

This function must be called before any attempt to perform a transformation or option setting can be made. If TSInit succeeds, TSDeInit must be called regardless of any other API calls.

### Prototype

```
TSERR TSInit(
    LPTSINITPARAMS pParams,
    PTSHANDLE      phSession);
```

### Parameters

- pParams: Pointer to a data structure, TSINITPARAMS, that describes C-Stub initialization parameters.
- phSession: Pointer to a variable of type TSHANDLE. Upon successful return from TSInit, this variable will be set to the "session handle" to be used in subsequent calls to the SCCTS interface. Note that this handle must not be used simultaneously in multiple threads.

### Return Values

- TSERR_OK: Indicates success.
- TSERR_UNKNOWN: Error trapping caught an unforeseen exception, such as an operating system or hardware exception. Please examine the status of the machine running the sccts module (including available memory, hard drive space and connectivity).

## TSINITPARAMSVER2 Structure

This structure is used to describe C-Stub initialization parameters.

This structure replaces the older TSINITPARAMS structure. While still valid, the TSINITPARAMS structure does not support client-side redirected IO on Linux systems where client and server reside on different machines. This new structure does support such configurations. It should also be noted that the newer structure resolves an issue seen in previous releases that affected developers on multi-homed machines.

### Structure

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

```
typedef struct TSINITPARAMStagVer2
{
   VTDWORD    dwVersion;
   VTLPSTR    szServer;
   VTWORD     wPort;
   OpenIOProc openIO;
   VTWORD     wIOPort;
   VTLPSTR    szIOServer;
} TSINITPARAMSVER2, *PTSINITPARAMSVER2;
```

- dwVersion: Identifies the version of this structure being used. Always set this value to SCCTS_INITPARAMS_CURRENTVERSION.

- szServer: Null-terminated string that contains the address where Transformation Server is listening for transformation requests. May be either a host name or dotted-decimal IP address.
- wPort: Port number upon which Transformation Server has been configured to listen for connections.
- openIO: Points to an OpenIO function (see Handling Redirected IO for more information). May be set to null if redirected IO is not being used.
- wIOPort: Port number on the local machine to be used for IO-related TCP communication between Transformation Server and sccts. If redirected IO is not used, this parameter is ignored. If redirected IO is used and this parameter is set to zero, an available port will be arbitrarily chosen for the IO communication.
- szIOServer: Host name/IP address of where redirected IO is taking place, needed only when the source or sink uses redirected IO. This is used in conjunction with the wIOPort parameter (null-terminated).

**Note:**

Redirected IO occurs when the specType field of a TS_IOSpec structure is set to the value of "redirect". Setting specType to "redirect" means that the file referenced in the TS_IOSpec structure needs to be accessed via client supplied IO functions. The wIOPort and szIOServer parameters are ignored if the OpenIO parameter is null.

### Example

```
TSINITPARAMSVER2   initParms;
initParms.dwVersion = SCCTS_INITPARAMS_CURRENTVERSION;
initParms.szServer = "server1.example.com";
initParms.wPort = 747;


if( TSERR_OK != TSInit(&initParms) )
{
   /* exit with an error */
}
```

## TSMemFree

This function allows the application using SCCTS to free allocated memory that was returned through the SCCTS interface. This function is not a generic de-allocator, and should only be called to free memory returned from an SCCTS interface function. Upon return from this function, the memory location pointed to by pvMem will be invalid.

### Prototype

```
TSMemFree (
   TSHANDLE    hSession,
   void* pvMem);
```

### Parameters

- hSession: The handle for the Transformation Server session.

- pvMem: A pointer to allocated memory returned from the SCCTS interface.

## TSSetOption

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

For **HTML Export** and **XML Export**: The identifier (hOptions) indicates what particular option is being specified. sccts supports two type of identifiers: predefined numeric ID values or text names. Numeric option identifiers may only be used to specify options for the Outside In Transformation Engines; all other transformation engines must use names for option identifiers. An option name may be any character string; its character set must be UTF-8 encoded Unicode. The names used for option identifiers are defined by the Export Engine.

The option value data (pOptionValue) must be in a form that conforms to the set of supported data types (see C/C++ Client Data Types ).

### Prototype

```
TSERR TSSetOption(
   TSHANDLE    hSession,
   VTLPSTR     szOptionName,
   VTLPVOID    pOptionValue,
   VTDWORD     dwOptionType
   );
```

### Parameters

- hSession: The handle for the Transformation Server session.
- szOptionName: Name of the option to be set in Unicode, null-terminated.
- pOptionValue: Pointer to the value of the option.
- dwOptionType: Type identifier indicating what type of option value is pointed to by pOptionValue. Allowable option types and their identifiers are described later in this book.

### Return Values

- TSERR_OK: Indicates success.
- TSERR_BADPARAM: One of the parameter values in the function call is incorrect.
- TSERR_UNKNOWN: Error trapping caught an unforeseen exception, such as an operating system or hardware exception. Please examine the status of the machine running the sccts module (including available memory, hard drive space and connectivity).
- TSERR_BADOPTIONTYPE: dwOptionType was invalid.

## TSSetOptionById

This function is called to set the value of a transformation option. It is provided to ease compatibility with code that was written to consume the embedded versions of Outside In Export Technologies. The options set-able by this function are specific to Outside In Export Technologies.

### Prototype

```
TSERR TSSetOptionById(
   TSHANDLE   hSession,
   VTDWORD    dwOptionId,
   VTLPVOID   pOptionValue,
   VTDWORD    dwOptionSize
   );
```

### Parameters

- hSession: The handle for the Transformation Server session.
- dwOptionId: The identifier of the option to be set.
- pOptionValue: Pointer to a buffer containing the value of the option.
- dwOptionSize: The size in bytes of the data pointed to by pOptionValue. For a string value, the null terminator should be included when calculating dwOptionSize.

### Return Values

- TSERR_OK: Indicates success.
- TSERR_BADPARAM: One of the parameter values in the function call is incorrect.
- TSERR_UNKNOWN: Error trapping caught an unforeseen exception, such as an operating system or hardware exception. Please examine the status of the machine running the sccts module (including available memory, hard drive space and connectivity).
- TSERR_INVALIDOPTION: dwOptionId was invalid.

## TSRunTransform

TSRunTransform is used to send the transformation request to the server.

### Prototype

```
TSERR TSRunTransform(
   TSHANDLE              hSession,
   LPTS_IOSpec           pSource,
   LPTS_IOSpec           pSink,
   VTLPSTR               szOutputFormat,
   VTLPSTR               szOptionSet,
   TS_TransformResult ** ppResults);
```

### Parameters

- hSession: The handle for the Transformation Server session.
- pSource: Pointer to a TS_IOSpec structure defining the transformation source. The valid values for this parameter are defined in the server-side configuration file, agent_iospec_types.xml. If unedited, this file specifies path, url or riot as the three valid values. See Specifying Inputs and Outputs with TS_IOSpec for more details.
- pSink: Pointer to a TS_IOSpec structure defining the transformation sink. The valid values for this parameter are defined in the server-side configuration file,

agent_iospec_types.xml. If unedited, this file specifies path, url or riot as the three valid values. See Specifying Inputs and Outputs with TS_IOSpec.

- szOutputFormat: Output format name in UTF-8 encoded Unicode, null-terminated. The valid output formats are contained in the configuration file called agent_engine_list.xml.
- szOptionSet: Option set name in UTF-8 encoded Unicode, null-terminated. Option sets can be coded by hand, or using the included option set editor (see The Option Set Editor).
- ppResults: Results of the transformation. It should also be noted that the data returned in the TS_TransformResult pointer must be freed by the calling application. This is done through a single call to TSMemFree. See TSMemFree for details.

### Return Values

- TSERR_OK: Indicates success.
- TSERR_BADPARAM: One of the parameter values in the function call is incorrect.
- TSERR_UNKNOWN: Error trapping caught an unforeseen exception, such as an operating system or hardware exception. Please examine the status of the machine running the sccts module (including available memory, hard drive space and connectivity).
- TSERR_OBJECTREFERENCEINVALID: AddReference method of the Options object returned NULL. Check that the Options parameter is valid.
- TSERR_OUTPUTFORMAT_NOTSUPPORTED: The output format designated in the call is not supported for the transform being attempted.
- TSERR_OPTIONSET_NOTSUPPORTED: The option set designated in the call is not supported for the transform being attempted.
- TSERR_ALLOCFAILED: Memory allocation failed.
- TSERR_INPUTOPENFAILED: Could not open the designated input file.
- TSERR_OUTPUTOPENFAILED: Could not create the designated output file.
- TSERR_FILECLOSEFAILED: A resource failed to be closed properly.

## TSDeInit

After the client application is done with all the possible transformations it needed to perform, TSDeInit needs to be called. This function should be called right before the sccts module is ready to be released. Make sure that this function is called in tandem with TSInit initialization function.

```
Prototype
TSERR TSDeInit(
   TSHANDLE hSession
   );
```

### Parameters

- hSession: The handle for the Transformation Server session.

#### Return Values

- TSERR_OK: Indicates success.
- TSERR_UNKNOWN: Error trapping caught an unforeseen exception, such as an operating system or hardware exception. Please examine the status of the machine running the sccts module (including available memory, hard drive space and connectivity).

## Sample Applications

Transformation Server ships with two sample applications that demonstrate the use of the SCCTS module: TSCLIENT and TSDEMO.

Some notes concerning these sample applications:

- The source code for these applications is provided, along with Microsoft Developer Studio project files. These applications were designed to demonstrate the use of the Transformation Server and the SCCTS module. They are freely modifiable, but are not warranteed and should not be assumed to be of commercial quality.
- TSCLIENT makes use of Microsoft Foundation Classes (MFC) and therefore must be built using Microsoft Developer Studio.
- Both of these applications demonstrate the use of custom IO Providers (also referred to as redirected IO). In both cases, the redirected IO code does nothing more than implement a thin layer on top of the file system input and output. As a result, the file specifications for these examples of redirected IO are identical to normal file system paths (unless the URL IO type is chosen, in which case the file specification must be a URL, not a file system path).
- In order to transform documents, both of these applications require Transformation Server (tsmanager) to be running and accessible via TCP/IP.
- These applications must be linked with SCCTS, which in turn requires access to various support DLL files located in the root level of the Outside In Transformation Suite directory. Running these sample applications from any other directory will result in an error, unless you add Transformation Suite's root directory to your path.
- The following information is needed to build the sample app tsdemo on Unix.

  To build on Solaris:

  ```
  make _solaris=1 clean all
  ```

  To build on Linux:

  ```
  make _linux=1 clean all
  ```

### tsclient

This is a Win32 application written in C++, with a dialog-based interface. Various controls and dialog boxes allow you to specify the options that affect how a document is transformed.

## tsdemo

> **Note:**
>
> To build tsdemo, you will need to link to lib/sccts.lib and include common/* in your path.

This is a command-line application written in C. The options that affect the transformation itself are controlled via a text-based configuration file. The command line parameters include the TCP host and port where Transformation Server is running, as well as the input, output, and configuration files to be used.

The command-line syntax is as follows:

```
tsdemo ServerName PortNumber InputFile OutputFile ConfigurationFile [IO Type
switch] [IOServerName]
```

Here is a guide to the valid command-line parameters for tsdemo:

- ServerName: The host name or IP address where Transformation Server is running. (Required)
- PortNumber: The port number on which Transformation Server will accept connections. (Required)
- InputFile: The path/URL to the input file. This parameter requires either an absolute path to a file, or a path to a file that is relative to Transformation Suite's root install directory. (Required)
- OutputFile: The path/URL to the output file. (Required)
- ConfigurationFile: The desired output format. (Required)
- IO Type: The type of IO specification used for the input and output parameters. May be a file system path (p), redirected (r), or URL (u). Default is p. (Optional)

> **Note:**
>
> Note for HTML Export: Even when u is specified for this parameter, tsdemo still expects a file path for the template and not a URL path.

- IOServerName: Host name/IP address of where redirected IO is taking place, needed only when the source or sink uses redirected IO. (Optional)

# 5

# Initiating Transformations Using the Java API

This chapter describes the use of the Java API. The Java API consists of several JAR files that need to be included in any third-party product. These files include the API JAR file, as well as several JAR files needed by the API. Example programs are supplied with the API to demonstrate the use of various Transformation Server features, but are not required to be included with any third-party applications.

Please note that you should be running Java version 1.6 or higher if you plan to use the Java API for Transformation Server. Also just to note, GNU java (gij) is not supported.

This chapter includes the following sections:



## Key Packages

The Java API for Transformation Server consists of classes from several packages. The packages are as follows:

- com.outsideinsdk.tsapi.api: Contains the main API classes, which all applications will need.
- com.outsideinsdk.tsapi.api.option: Contains classes used for setting common options in Transformation Server.
- com.outsideinsdk.tsapi.api.option.hwx: Contains classes used for setting options related to the Export engine.
- com.outsideinsdk.tsapi.api.option.xsd: Contains classes used for setting common SOAP options.
- com.outsideinsdk.tsapi.api.message: Contains classes used to represent messages returned from Transformation Server.
- com.outsideinsdk.tsapi.api.redirect: Contains classes used for redirected IO.

## Key Classes

While the Java API incorporates many classes, there are some classes that developers will need to use more than others, and therefore should be more familiar with.

- com.outsideinsdk.tsapi.api.TransformClient: This class is the bulk of the API. A connection to an instance of Transformation Server is created by instantiating an instance of this class. All calls to the server go through this class. Every Transformation Server application will need at least one instance of TransformClient.
- com.outsideinsdk.tsapi.api.option.xsd: This class is the base class of all option classes. Applications won't instantiate this class directly, but through the use of other option classes.
- com.outsideinsdk.tsapi.api.option.xsd.BooleanOption

  com.outsideinsdk.tsapi.api.option.xsd.FloatOption

  com.outsideinsdk.tsapi.api.option.xsd.IntegerOption

  com.outsideinsdk.tsapi.api.option.xsd.LongOption

  com.outsideinsdk.tsapi.api.option.xsd.StringOption

  com.outsideinsdk.tsapi.api.option.xsd.UnsignedIntOption

  These classes are used as substitutes for native objects when used as option values.

  The UnsignedIntOption class is a wrapper for a Long value. The reason for this is that several options in Transformation Server require unsigned integers as values, but Java does not support this type. For this reason, the Java API reads the value and interprets it as the larger Long type.
- com.outsideinsdk.tsapi.api.option.StringData: The StringData class is different than the StringOption class. It includes a field for the character set to be specified, and also allows the actual bytes storing the string to be Base64 encoded. This class is used many times in the Java API.
- com.outsideinsdk.tsapi.api.option.IOSpec: This class is used for all file specifications. It consists of a StringData object specifying the path to the file, and a string that specifies the path type. Currently supported path types are path and redirect. IOSpec objects with a path type of path are expected to be accessible locally by Transformation Server, while IOSpec objects with a path type of redirect use redirected IO to allow the server to access files on the client machine.
- com.outsideinsdk.tsapi.api.redirect.BaseIO: This interface is used to allow the Java API to interact with redirected IO sources. Through this interface, developers provide a common interface to their data so that the Java API can perform any required IO operations. A default implementation is provided in the example code.
- com.outsideinsdk.tsapi.api.redirect.OpenIO: This class is the interface through which the Java API is able to open files for redirected IO. Any third-party implementation that wishes to incorporate redirected IO must implement this interface. A default implementation is provided in the example code.
- com.outsideinsdk.tsapi.api.redirect.IOProvider: This class is used as a registry for the OpenIO class. When an application is using redirected IO, the implementation of the OpenIO interface must be registered with the IOProvider class through the static register() method.

## Redirected IO

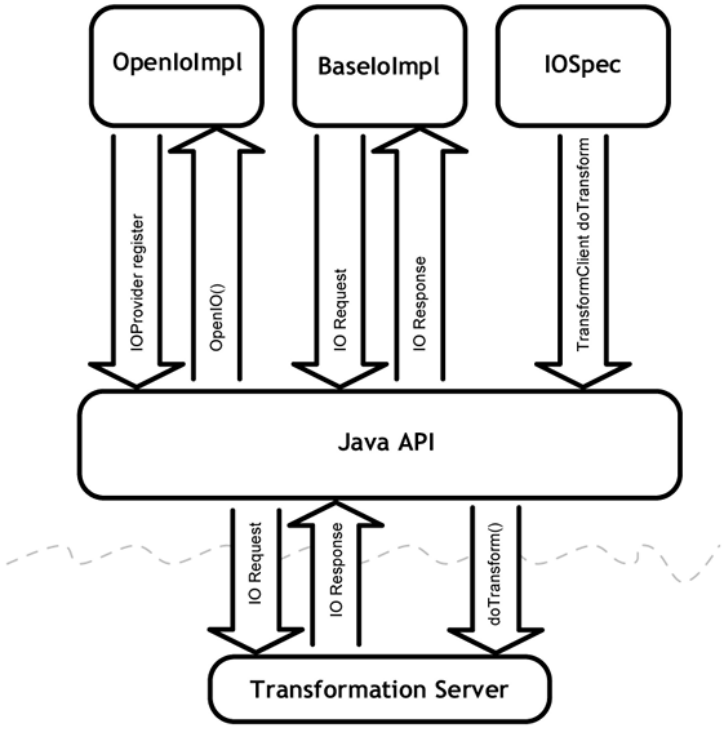
Transformation Server allows developers to specify how IO should be performed. Currently, Transformation Server handles three types of IO natively; file system, URL

and redirected IO. When specifying the path using the type path, Transformation Server treats the file specification as a path to a file on the local file system. When specifying the path using the type url, Transformation Server treats the file specification as an Internet address. A path type of redirect allows the developer more control over the IO functions.

Redirected IO allows file IO operations to be done remotely. Further, redirected IO allows the developer to control all aspects of the actual IO operations. Through the use of the BaseIO and OpenIO interfaces, developers can take control of all IO operations. To accomplish this, developers need to supply the Java API with an implementation of the OpenIO interface. This is achieved by calling IOProvider.register(), and passing it an instance of the OpenIO implementation.

When the Java API receives a request for an IO operation, it first checks to see if the file has been opened. If it has not, it calls the openIO() method in the OpenIO implementation, passing it the IOSpec and an OpenIOFlag object. This method is expected to return an instance of a BaseIO implementation that will allow the Java API to perform all necessary IO operations. Once the file has been opened, all communication occurs directly between the Java API and the BaseIO implementation. See the following diagram for further explanation.

***Figure 1 Java API***




# Sample Applications

The following sections contain sample applications.

## TSJavaDemo

The class *c*om.outsideinsdk.tsapi.example.TSJavaDemo is a sample application that uses several helper classes to demonstrate the uses of the Transformation Server Java Client API. It takes several parameters from the command line and optionally reads a text-based file containing transformation options. These are then passed to

Transformation Server via the Java Client API to execute the transformation. A batch file called tsjavademo (located in sdk**[platform name]**\sdk\samplecode \tsjavademo) is provided for convenient access to this application.

This application simply creates a TransformClient object, calls doTransform() on that object, and then prints the result. While this is a very simple example, it clearly shows the required steps for executing a transformation using Transformation Server's Java API.

Please note that all path types referred to in the following documentation are dependent on the setting for the -t option. The default value for the -t parameter is p, meaning that paths are relative to the tsmanager executable on the machine which is hosting it.

The available command line parameters are as follows:

- -s, -server : The host name or IP address where Transformation Server is running. (Required)
- -p, -port: The port number on which Transformation Server will accept connections. (Required)
- -i, -input: The path/URL to the input file. This parameter requires either an absolute path to a file, or a path to a file that is relative to Transformation Suite's root install directory. To use an input path relative to the tsjavademo sample app, the -t r (redirected IO) option should be used. (Required)
- -o. -output: The path/URL to the output file. (Required)
- -r, -format: The desired output format. This value comes from the agent_engine_list.xml file in the root level of the install directory. (Required)
- -f, -optionFile: The option file, which can be a *.cfg file, readable by tsdemo, or an XML file. This file must be relative to tsjavademo on the computer sending the request. (Optional)
- -n, -optionSetName: The option set from the file in -f, if it is an XML file. (Optional)
- -t, -ioType: The type of IO specification used for the input and output parameters. May be a file system path (p), redirected (r), or URL (u). Default is p. Note for HTML Export: Even when u is specified for this parameter, tsjavademo still expects a file path for the template and not a URL path. This path is relative to tsmanager on the computer receiving the request. (Optional)
- -h, -help: Shows this list of parameters.

### Notes on the Sample Application

- The source code for this application is provided. It is designed to demonstrate the use of the Transformation Server and the SCCTS module. It is freely modifiable, but is not warranteed and should not be assumed to be of commercial quality.
- This application demonstrates the use of custom IO Providers (also referred to as redirected IO). The redirected IO code does nothing more than implement a thin layer on top of the file system input and output. As a result, the file specifications for this example of redirected IO is identical to normal file system paths (unless the URL IO type is chosen, in which case the file specification must be a URL, not a file system path).

- In order to transform documents, this application requires Transformation Server (tsmanager) to be running and accessible via TCP/IP.

## URL Input and Output

When the paths for input and output files are specified as URLs (ioType parameter equals *u*), the output files will be sent to the output URL via the HTTP PUT command. In order for this to work correctly, the Web server hosting the output URL must be configured to allow writing to the output URL path.

## Redirected Input and Output

When the paths for input and output files are specified as "redirected" (ioType parameter equals r), the sample application will demonstrate how custom objects can be used in place of system calls to provide reading and writing of input and output files. To do so, the developer must implement the BaseIO and OpenIO interfaces.

The sample BaseIO implementation is the following:

```
com.outsideinsdk.tsapi.example.BaseIOExample
```

The sample OpenIO implementation is the following:

```
com.outsideinsdk.tsapi.example.OpenIOExample
```

These two interface implementations demonstrate how developers can implement redirected IO in their own applications. For the Java Client API to make use of these objects, a call is made to IOProvider.register() (registering the OpenIOExample implementation class) and the IOSpec spec types are set to redirect.

# 6

# Transformation Engine Specification

This chapter describes the Transformation Engine, which is a module loaded by a Transformation Agent in order to transform a document. To communicate with the Transformation Agent, a Transformation Engine must implement a relatively simple API to support setting of options and executing a transformation.

A transformation engine is not expected to understand SOAP or handle any interprocess communication (except possibly for private reasons). Integration with the larger Transformation Server architecture is the responsibility of the agent that hosts the transformation engine, not the engine itself.

An engine must provide an entry point function named LoadEngine which is used to initialize the engine itself and its communication with the agent. The engine must also implement a handful of functions that are used by the agent to control the process of a transformation. These functions are called the "engine interface." The agent also supplies a set of functions, called the "agent interface," to be used by the engine as needed during a transformation.

The engine and agent present their interfaces to each other through C data structures that contain pointers to functions. With the exception of the LoadEngine entry point, all communication between the engine and the agent is accomplished through the function pointers in these data structures.

This chapter includes the following sections:



## Getting Started

The following section describe basic steps in getting started.

### Transformation Engine Interface

Transformation Engines are implemented as loadable modules that can be hosted by any Transformation Agent. Currently, Transformation Engines must implement a C-language API to be successfully integrated into a Transformation Agent. The Transformation Engine interface itself is very simple: it consists of a mapping of the Transform API, and the means to access Transformation Server's IO interface for the inputs and outputs of a transformation.

### Loading Mechanism

An engine must provide an entry point function that is called by the Transformation Agent to initialize a transformation. This entry point is used to exchange data structures containing pointers to all of the other API functions on both sides of the interface. One data structure, set by the agent, contains pointers to the functions within the agent that can be called by the engine (the "engine-to-agent" interface). The other data structure contains pointers to the "agent-to-engine" functions, which must be provided by the engine for use by the agent.

A Transformation Agent determines which Transformation Engine to use for a given transformation request by inspecting its configuration information – each supported output format must be mapped to a Transformation Engine. This mapping may be extended with new output formats and/or Transformation Engines at any point in time.

### The Agent-to-Engine Interface

The agent-to-engine interface consists of four functions:

- openTransform: Notify an engine to start transforms.
- setOption: Set options for the transform.
- transform: Perform the transform.
- closeTransform: Release resources after a transform.

See Engine Interface for details.

### The Engine-to-Agent Interface

The engine-to-agent interface currently consists of four functions:

- openIO: Engine notifies the agent of IO Provider API ioOpen calls.
- addToOutputList: Engine notifies the agent of additional files that were created.
- setResultMsg: Engine notifies the agent of custom result messages.
- logMessage: Engine notifies the agent of messages to be sent to the message logger.

See Agent Interface for details.

The openIO function provides Transformation Engines with a means to access the input and output documents through a BASEIO object (see the description of the IO Provider interface in the next section). Calling the function will cause the Transformation Agent to load an appropriate IO Provider for the IO specification specified. The agent will retrieve a BASEIO printer from the IO Provider, and pass it back to the engine.

Engines are not required to use BASEIO objects for input and output if they are able to open, read, and/or write the specified input and output documents through other means (such as the file system). However, and engine that does access its input and output documents via the BASEIO object will benefit from any IO providers that are installed on the server or provided remotely from the client.

All of the Outside In transformation engines will access input and output documents exclusively through the IO Provider interface.

### Required Header Files

A Transformation Engine must have access to all of the Transformation Server header files, and it must include TS_ENGINE.H. This file will define the prototype of the entry point function, the structures for the interface, and will pull in any additional header files needed by Transformation Server.

### Transformation Agent Configuration

Once an engine has been built, the Transformation Agent must be configured to know where to find the engine. This is done by modifying a configuration file named agent_engine_list.xml. This file contains a list of transformation engines and the formats each one supports.

## Transformation Engine Entry Point

The following is entry point information.

### LoadEngine

The LoadEngine function is the entry point through which the Transformation Agent initiates and terminates a conversation with a Transformation Engine. This function is called once immediately after the engine is loaded, and once immediately prior to unloading.

On loading, this function the engine will verify that it supports the specified version of the interface, and if so it will initialize the rest of the EngineInterface structure. On unloading, the engine may perform any cleanup tasks it requires.

#### Prototype

```
TSERR LoadEngine( EngineInterface * pEngine, bool bLoading )
```

#### Return Values

- TSERR_OK: If the function is successful
- TSERR_ENGINEVERSION: If the engine does not support the version reported by the Engine structure. In this case, the engine should set its preferred EngineInterface version number in the version field of the EngineInterface structure.
- …other TSERR values: As needed.

#### Parameters

- pEngine: This parameter is a pointer to an EngineInterface structure that should be filled in by the transformation engine. It contains pointers that should be set to point to their corresponding functions inside the engine module.
- bLoading: This parameter is true if the engine has just been loaded, or false if the engine is about to be unloaded.

## Engine Interface

The following sections describe the engine interface.

## EngineInterface Structure

The TransformationEngine structure is defined as follows:

```
typedef TSERR (*TRANSFORMPROC)(TS_IOSpec *src, TS_IOSpec *sink,
    VTLPVOID pEngineData, AgentInterface * agent);
typedef TSERR (*OPENTRANSFORMPROC) (TS_char * outputType,
    void * * pEngineData);
typedef void (*CLOSETRANSFORMPROC)(VTLPVOID pEngineData);

typedef struct EngineInterface
{
    XSD_unsignedInt    version;
    OPENTRANSFORMPROC  openTransform;
    TRANSFORMPROC      transform;
    SETOPTIONPROC      setOption;
    CLOSETRANSFORMPROC closeTransform;

} EngineInterface;
```

- version: This specifies the version of the transformation engine API specification to which this engine was written. The format of this number is not currently documented, but later versions are guaranteed to have a higher version number than earlier versions. Developers should set this value to kTransformEngineInterfaceVersion, which is the current version as defined by the header files in use.
- openTransform: This is a pointer to the transformation engine's openTransform function
- transform: This is a pointer to the transformation engine's transform function
- setOption: This is a pointer to the transformation engine's setOption function
- closeTransform: This is a pointer to the transformation engine's closeTransform function

## openTransform

The openTransform function is the entry point through which the Transformation Agent initiates a conversation with a Transformation Engine. This function is called each time a new transformation operation is about to begin.

### Prototype

```
TSERR openTransform(TS_char * outputFormat, void **
    pEngineData);
```

### Return Values

- TSERR_OK: If the function is successful
- TSERR_FORMATNOTSUPPORTED: If the engine does not support the specified output format.:
- …other *TSERR* values: as needed

### Parameters

- outputFormat: This identifies the selected output format for the transformation. This is a null-terminated string, composed of UTF-8 encoded Unicode characters.
- pEngineData: The value pointed to by pEngineData should be set to a value for the private use of the engine. This value will be passed back to the engine on subsequent interface calls. Typically, an engine would set *pEngineData to point to some data structure that it uses for tracking data specific to the current transformation.

## setOption

This function sets options for transformations.

### Prototype

```
TSERR setOption(TS_char * name, void * data, XSD_unsignedInt
    type, void * engineData);
```

In practice, it is possible that setOption will be called more than once for the same option. By definition, the last value set for an option should be considered its "true" value.

### Return Value

Returns TSERR_OK if successful, or an error if the option could not be set.

### Parameters

- name: The name of the option being set
- data: This is a void pointer to the value associated with the option being set.
- type: This identifies the type of the data pointed to by the value pointer. Possible values are any of the following:
  - XSD_boolean_type
  - XSD_string_type
  - XSD_float_type
  - XSD_double_type
  - XSD_int_type
  - XSD_short_type
  - XSD_byte_type
  - XSD_unsignedInt_type
  - XSD_unsignedShort_type
  - XSD_unsignedByte_type
  - TS_CharacterSetEnum_type
  - TS_stringData_type

- TS_IOSpec_type
- TS_binaryData_type
- OIT_AltLink_type
- OIT_CellHeadings_type
- OIT_CharMappingEnum_type
- OIT_CharacterAttributes_type
- OIT_CharacterByteOrderEnum_type
- OIT_ComplianceEnum_type
- OIT_DatabaseFitToPageEnum_type
- OIT_DefaultFont_type
- OIT_DefaultInputCharSetEnum_type
- OIT_DefaultMargins_type
- OIT_DefaultPageUnitsEnum_type
- OIT_DocumentMemoryModeEnum_type
- OIT_EmailHeaderOutputEnum_type
- OIT_ExtractEmbeddedFilesEnum_type
- OIT_FlavorEnum_type
- OIT_FallbackFormatEnum_type
- OIT_FontFlags_type
- OIT_GraphicCroppingEnum_type
- OIT_GraphicSizeMethodEnum_type
- OIT_GraphicTypeEnum_type
- OIT_GraphicWatermarkScaleTypeEnum_type
- OIT_GridAdvanceEnum_type
- OIT_MimeHeaderOutputEnum_type (deprecated)
- OIT_ParagraphAttributes_type
- OIT_ReorderMethodEnum_type
- OIT_SearchMLFlags_type
- OIT_SearchMLUnmappedTextEnum_type
- OIT_SpreadsheetFitToPageEnum_type
- OIT_SpreadsheetPageDirectionEnum_type
- OIT_SpreadsheetShowBorderEnum_type

- OIT_TiffOptions_type
- OIT_WatermarkPositionEnum_type
- OIT_WatermarkScalingEnum_type
- OIT_XmlDefinitionMethodEnum_type

- engineData: This is the engineData value supplied by the transformation engine during the OpenTransform function.

### transform

This is the function that actually accomplishes a transformation.

#### Prototype

```
TSERR transform(struct TS_IOSpec *src, struct TS_IOSpec *sink,
    void * engineData, AgentInterface * agent);
```

#### Return Value

Returns TSERR_OK if successful, or an error if the transformation failed.

#### Parameters

- src: This is the IO specification of the input document for the transformation.
- sink: IO specification for the output of the transformation
- engineData: This is the engineData value supplied by the transformation engine during the OpenTransform function.
- agent: The pAgent parameter is a pointer to an AgentInterface structure, through which the engine can communicate back to the Agent.

### closeTransform

This function is called by the transformation agent to notify the transformation engine that the transformation represented by hTransform may be disposed.

```
void closeTransform(void * hTransform);
```

#### Parameters

- hTransform: This is the identifier supplied by the transformation engine as the return value of the OpenTransform function.

## Agent Interface

The following information pertains to agent interface features.

### AgentInterface Structure

The AgentInterface structure has the following definition:

```
typedef TSERR (*OPENIOSPECPROC)(TS_IOSpec * spec,
    XSD_unsignedInt flags, BASEIO ** ppDoc, struct
    AgentInterface * agent );
typedef TSERR (*ADDTOOUTPUTLISTPROC)(TS_char * spec,
```

```
    TS_CharacterSetEnum charSet, TS_char * specType,
    AgentInterface * agent );
typedef TSERR (*SETRESULTMSGPROC)(TS_char * spec,
    TS_CharacterSetEnum charSet, struct AgentInterface * agent
    );
typedef TSERR (*LOGMESSAGEPROC)(TS_char * msg,
    TS_CharacterSetEnum charSet, TS_MessageTypeEnum type, struct
    AgentInterface * agent );

typedef struct AgentInterface
{
    XSD_unsignedInt      version
    OPENIOSPECPROC       openIO;
    ADDTOOUTPUTLISTPROC  addToOutputList;
    SETRESULTMSGPROC     setResultMsg;
    LOGMESSAGEPROC       logMessage;
} AgentInterface;
```

- version: This specifies the version of the transformation engine API specification in use by the transformation agent host. This is the same value as described for the EngineInterface version field.
- openIO: Points to the agent's openIO function.
- addToOutputList: Points to the agent's addToOutputList function
- setResultMsg: Points to the agent's setResultMsg function
- logMessage: Points to the agent's logMessage function

## openIO

This function is called by the engine to open an "IO object", which is a source of input or destination of output. Access to the IO object is provided through a BASEIO structure.

```
TSERR openIO(TS_IOSpec * spec, XSD_unsignedInt flags, BASEIO **
    ppDoc, AgentInterface * agent)
```

### Parameters

- spec: A pointer to a TS_IOSpec structure that specifies a document to be opened for reading or writing
- flags: One or more flags that indicate how the document is to be opened. Possible values are:
  - IOOPEN_READ: The document is being opened for reading
  - IOOPEN_WRITE: The document is being opened for writing
  - IOOPEN_CREATE: The document is being created. If a document of the same name exists, it will be replaced by the new document. Any documents opened with the IOOPEN_CREATE flag will automatically be added to the list of output documents reported with the results of the transformation, unless IOOPEN_PRIVATE is also specified.
  - IOOPEN_PRIVATE: When use with IOOPEN_CREATE, prevents the file from being included in the list of output files for the current transformation.
- ppDoc: If the openIO function is successful, this variable will be set to point to a BASEIO structure that will provide access to the document that was opened.

- agent: The pointer to the AgentInterface structure that was passed as a parameter to the transform function

#### Return Values

TSERR_OK if the operation was successful, or an error value if it was not.

## addToOutputList

This function is called by the engine when creating output documents through some means other than the openIO function. Documents specified through this function will be included in the list of output documents that is reported to the originator of the transformation request.

Any documents created via the agent's openIO function are automatically added to the output list; for those documents there is no need to call this function.

#### Prototype

```
TSERR addToOutputList(TS_char * spec, TS_charsetEnum charSet,
    TS_char * specType, AgentInterface * agent )
```

#### Return Values

TSERR_OK if the operation was successful, or an error value if it was not.

#### Parameters

- spec: The specification of the output document
- charSet: The character set used in the spec string
- specType: The type of specification (for example, "path", "URL", etc.)
- agent: The pointer to the AgentInterface structure that was passed as a parameter to the transform function

## setResultMsg

This function is called by the engine to specify a result string for the transformation operation. Use of this function is optional.

```
TSERR setResultMsg(TS_char * msg, TS_charsetEnum charSet,
    AgentInterface * agent )
```

#### Return Values

TSERR_OK if the operation was successful, or an error value if it was not.

#### Parameters

- msg: A string containing the result message
- charSet: The character set used in the message string
- agent: The pointer to the AgentInterface structure that was passed as a parameter to the transform function

## logMessage

This function is called by the engine for diagnostic purposes, to add a string to Transformation Server's error log.

```
TSERR logMessage(TS_char * msg, TS_CharacterSetEnum charSet,
    TS_MessageTypeEnum type, struct AgentInterface * agent );
```

### Return Values

TSERR_OK if the operation was successful, or an error value if it was not.

### Parameters

- msg: A string containing the error message
- charSet: The character set used in the message string
- type: One of the following:
  - msgError: The message describes an error
  - msgInfo: The message is for informational purposes
  - msgStatus: The message provides ongoing status information about the current transformation
- agent: The pointer to the AgentInterface structure that was passed as a parameter to the transform function

# 7

# IO Provider Specification

This chapter describes the IO Provider. In order to support the reading, writing, and creation of documents that are not stored on the operating system's file system, Transformation Server has defined a generic input/output interface called the IO Provider Interface. An **IO Provider** is a module that implements this interface. On the other side of this interface, the code that loads and uses an IO Provider is called an **IO Consumer**. All document access in Transformation Server is accomplished through the IO Provider interface; it is also available for use by any third-party developer of a custom transformation engine.

This version of the IO Provider interface is available only for developers coding in C or C++. Users writing code in Java who wish to perform redirected IO should refer to Redirected IO.

The IO Provider Interface allows bi-directional random access to a stream of data, and is modeled after file-based input and output. (We also refer to this functionality as "redirected IO" because the transformation process is operating on a source other than a file.) This interface allows an IO Consumer to treat any target data stream as if it were a file, while leaving the IO Provider itself responsible for the specific details of accessing and modifying that data stream.

The IO Provider Interface includes operations for creation, opening, closing, reading, writing, and seeking; as well as a method for querying the IO Provider for various data about the particular IO target.

An IO Provider acts as a "plug-in" component to Transformation Server. An IO Provider does not implement SOAP or any other type of interprocess communication (except possibly for private reasons). Integration with the Transformation Server infrastructure and the engine that accomplishes a transformation is handled by Transformation Server itself.

This chapter includes the following sections:



## IO Provider Interface

The IO Provider interface is a generic means of opening, creating, reading or writing documents, modeled on file-based input/output functions. This interface may be implemented by a developer to provide Transformation Server with access to documents that don't reside on a file system.

## Why Use IO Providers?

A developer writing an application that uses Transformation Server may wish to transform or create documents that reside in a database, document management system, some other repository, or a file system to which Transformation Server does not have access. If Transformation Server supported only file-based IO, these applications would be required to copy and manage temporary files as part of their interaction with Transformation Server.

The IO Provider interface supplies a more flexible solution to the problem. Once an IO provider is written for a particular repository, it can be used by any transformation technology installed on Transformation Server.

## IO Specifications

The specification of an input or output document is simply the identifier by which it is known in its native repository; for example, a file's specification is its path. Transformation Server requires that input and output documents be identified by both a specification and a specification type, which is a text label that identifies how the specification should be interpreted. The two specification types that are natively supported by Transformation Server are file-system paths and URLs, but any IO Provider can extend this set. The Transformation Agent configuration file maintains a mapping of each specification type to the IO Provider module that supports it.

An IO Provider may support any number of specification types, but only one IO Provider may handle any given specification type.

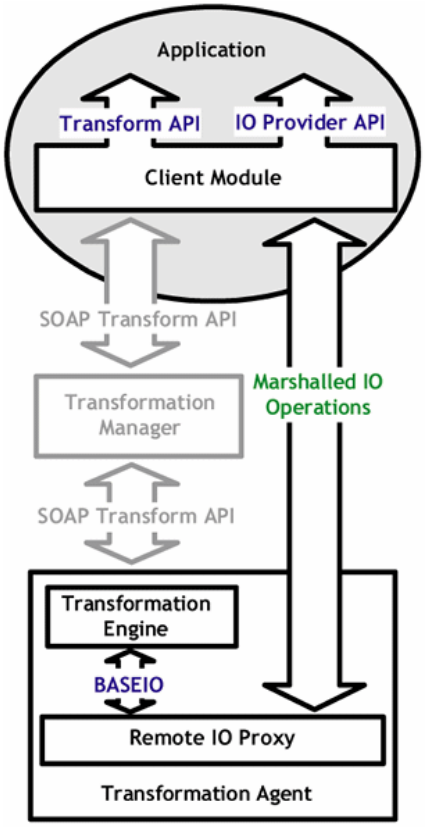
## Server-Side Versus Client-Side IO Providers

A "client-side" IO Provider executes its code in the same process as the client application, while a "server-side" IO Provider is supplied as a module that is loaded by a Transformation Agent when needed. Transformation Server defines two versions of the IO Provider interface: one for Java that is only available on the client-side, and a C-language version that can be deployed on either the client or server sides of Transformation Server.

Server-side IO is straightforward: the appropriate IO Provider module is loaded by the Transformation Agent and provides functions through which the Transformation Engine reads and writes its input and output.

Client-side IO is a little more complicated. For one thing, it is only supported when the application is using either the C/C++ or Java client modules to communicate with Transformation Server.

***Figure 1 Client Side IO Provider***




*Using a Client-Side IO Provider*

In order for the Transformation Engine on the server to read data from a client-side IO provider, the engine interacts with a proxy IO Provider (a component supplied with Transformation Server). The proxy communicates to the C/C++ or Java interface module, relaying the IO operations via a private protocol. The client module in turn uses the client-side IO Provider to perform the operation, the results of which are sent back up to the server-side proxy.

The IO communication flows through a different socket than the SOAP communication that controls the Transformation Manager, and won't interfere with the general operation of Transformation Server.

## The C Version

Most of the IO Provider functionality is encapsulated in the BASEIO data structure, which contains pointers to the IO functions implemented by the IO Provider. Transformation Engines interact directly with the BASEIO structure to open, create, read, and write input and output documents.

The Transformation Agent obtains a pointer to a BASEIO object from an IO provider by calling OpenIO, which is an entry point function that must be implemented by the IO Provider. This function receives the specification of the document to be opened/created, flags that tell how the document is to be opened, and a pointer to a structure that provides access to functions within the Transformation Agent.

The BASEIO structure contains pointers to functions that provide the following operations: read, write, seek, tell, close, and GetInfo. The first five functions are directly analogous to common file operations. The GetInfo function provides

information for various queries that aren't directly related to reading, writing, or positioning within the IO stream. The Transformation Engine may use the GetInfo function to determine things such as the size of the stream, a URL that should be used to link to the stream, or an IO specification for an additional output stream.

If a developer has written both a Transformation Engine and an IO Provider, the GetInfo function may also be used to exchange private data between them.

## The Java Version

The functionality of the Java version is identical to the C client version, except that the data structures and functions referred to in the previous section are replaced with similarly-named Java classes (BaseIO and OpenIO).

The basic architecture of the Outside In technology is the same across all supported platforms.

# Configuration

This section contains basic configuration information.

## Server-Side Versus Client-Side Operation

As a client-server application, Transformation Server runs its transformation operations in a separate process from the "client" application that uses it. For maximum flexibility, Transformation Server allows IO Providers to supply redirected IO on either the server side or the client side. In other words, the code that provides the IO may execute in the process of the client application or in the process where the transformation occurs. From an implementation point of view, there is little to no difference between the two approaches; it is entirely possible to use the same binary code to provide redirected IO on both the client and server.

In server-side redirected IO, the IO Provider must be built as a dynamically loadable library that is loaded by the transformation process as needed. Communication with the IO Provider then proceeds in-process as the transformation is being performed. This provides more efficient performance than client-side redirected IO, and should be considered the preferred configuration.

An application that wishes to use client-side redirected IO must also use the C Client interface (SCCTS module). The C client API will communicate privately with the server-side Transformation Agent to marshal all IO operations between the two processes.

### Installing an IO Provider on the Server

Build your IO Provider code into a loadable module/DLL, with OpenIO specified as an exported function. Copy this module to a location accessible to Transformation Server, such as the root level of the install directory.

Modify the Transformation Server configuration file agent_iospec_types.xml to indicate the location of your IO Provider. In this step, you must also define the name of an IO spec type that will be mapped to your IO Provider (the specType field in the TS_IOSpec structure). Note that the specType value redirect is reserved for client-side redirected IO, and is not allowed for a server-side IO provider.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<IoSpecTypeToModuleMap xsi:type="..." attributes deleted for
   readability>
```

```
      <SpecType xsi:type="tss:SpecType">
      <!-- For local file system and shared file system paths-->
      <Name xsi:type="xsd:string">myspectype</Name>
      <Module xsi:type="xsd:string">my_io_module.dll</Module>
   </SpecType>
   ...Other SpecType elements ...
</IoSpecTypeToModuleMap>
```

### Using an IO Provider on the Client

To use an IO Provider on the client, you supply the address of your OpenIO function as a parameter to the TSInit function in the SCCTS module. Then, when you need to supply a specification of a data stream that should be opened with your IO Provider, you use the reserved specType value redirect.

# IO Provider Entry Point

This section describes the OpenIO function, which is the entry point through which an IO Consumer opens a new data stream for reading or writing, and the BASEIO data structure.

## OpenIO

```
IOERR OpenIO( const * ioSpec,
   VTDWORD dwFlags,
   IOConsumerInterface * pConsumer,
   BASEIO ** ppBaseIO);
```

The OpenIO function is the entry point through which an IO Consumer opens a new data stream for reading or writing.

#### Parameters

- ioSpec: This is the specification of the data stream to be opened, or created and opened.
- dwFlags: The flags indicate whether the file is opened for read, write and create operations. These flags may be combined together to as desired. The valid flags are:
  - IOOPEN_READ: The file should be opened for read.
  - IOOPEN_WRITE: The file should be opened for write. Please note that if the specified file already exists, it's contents will be erased when this flag is set.
  - IOOPEN_CREATE: The file should be created and opened for write.
- pConsumer: This is a pointer to the IO Consumer interface of the caller.
- ppBaseIo: This is a pointer to a pointer to a BASEIO structure. The IO Provider must allocate and initialize this data structure, and set ppBaseIO to point to it. A new BASEIO structure must be separately allocated for each data stream opened via this call.

#### Return Values

This function returns an IOERR value:

- IOERR_OK: The IO Provider was able to initialize and open the data stream
- IOERR_NOFILE: The data stream could not be opened.

- IOERR_NOCREATE: The data stream could not be created.
- IOERR_BADPARAM: An invalid parameter was given to OpenIO.
- IOERR_INVALIDSPEC: The IOSpec not valid for this IO Provider.
- IOERR_FALSE: The reported version of the IOConsumerInterface is not supported.

## The BASEIO Structure

The BASEIO structure is the means through which an IO Provider allows an IO Consumer access to its interface functions. It is a data structure that contains pointers to each of the interface functions implemented by the IO Provider. A pointer to this data structure is also included as a parameter in each of the functions it provides.

The BASEIO data 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 fields of the BASEIO data structure should be set as follows:

- pClose: Set this to point to your IOClose function
- pRead: Set this to point to your IORead function
- pWrite: Set this to point to your IOWrite function
- pSeek: Set this to point to your IOSeek function
- pTell: Set this to point to your IOTell function
- pGetInfo: Set this to point to your IOGetInfo function
- pOpen: Reserved. Must be set to null.
- pSeek64: Set this to point to your 64-bit IOSeek function
- pTell64: Set this to point to your 64-bit IOTell function
- aDummy: Reserved

### Remarks

In practice, when implementing an IO Provider you will probably need to associate your own private data with the BASEIO structure in order to maintain state information during IO operations. (You will be handed back your BASEIO pointer in every IO operation.)

You may do this in C by defining your own data structure that includes a BASEIO structure as its first element.

For example:

```
typedef struct MyIOStruct
{
   BASEIO               baseIO;
   MYDATA               myDataStream;
   IOConsumerInterface  * pConsumer;
   ...etc...
} MyIOStruct;
```

Then, from your OpenIO function, you would return a pointer to this data structure, cast as a BASEIO pointer:

```
MyIOStruct * pMyData;
pData = (MyIOStruct *) malloc(sizeof(MyIOStruct));

/* open the data stream and initialize the struct, then... */

*ppBaseIO = (BASEIO *) pMyData;
```

# IO Provider Functions

For compatibility with older versions of Outside In SDKs, the type of the first parameter in the IO Provider functions (BASEIO *) may also be referred to by the typedef HIOFILE in header files and/or documentation. These two types should be considered interchangeable, as both are required to be the address of a BASEIO structure.

## IOClose

This function tells the IO Provider that the IO Consumer has finished using a pointer to a BASEIO structure. Any allocated resources associated with the BASEIO may now be released. The IO Consumer will not make any use of this BASEIO structure after calling IOClose.

```
IOERR IOClose(BASEIO* pBaseIO);
```

### Parameters

- pBaseIO: The BASEIO structure.

### Return Values

- IOERR_OK: The data stream was closed.
- IOERR_UNKNOWN: The data stream could not be closed.

## IORead

Reads the next *size* bytes from the current position in the data stream; the current position should then be set to the byte after the last byte read.

```
IOERR IORead(BASEIO* pBaseIO, VTLPBYTE pData, VTDWORD size,
   VTLPDWORD pCount);
```

### Parameters

- pBaseIO: A pointer to the BASEIO structure for this data stream.

- pData: Pointer to the buffer to read the data into.
- size: The number of bytes to be read.
- pCount: IORead sets *pCount to the number of bytes actually read.

#### Return Values

- IOERR_OK: Read was successful. pCount contains the number of bytes read and pData contains the bytes themselves. A request for 0 bytes should return an unknown error.
- IOERR_EOF: The data stream was already at the end of the file when this call was received.
- IOERR_UNKNOWN: The data stream could not be read.

## IOWrite

Writes *size* bytes at the current position in the data stream; the current position should then be set to the position after the last byte written.

```
IOERR IOWrite(BASEIO* pBaseIO, VTLPBYTE pData, VTDWORD size,
   VTLPDWORD pCount);
```

#### Parameters

- pBaseIO: A pointer to the BASEIO structure for this data stream.
- pData: Points to the data to be written to the data stream.
- size: The number of bytes to be written.
- pCount: IOWrite sets *pCount to the number of bytes actually written.

#### Return Values

- IOERR_OK: The data stream was successfully written.
- IOERR_UNKNOWN: The data stream could not be written.

## IOSeek

Moves the current data stream position.

```
IOERR IOSeek(BASEIO* pBaseIO, VTWORD wFrom, VTLONG lOffset);
```

#### Parameters

- pBaseIO: A pointer to the BASEIO structure for this data stream.
- wFrom: One of the following values:
  - IOSEEK_TOP: Move the data stream position to lOffset from the beginning of the data stream.
  - IOSEEK_BOTTOM : Move the data stream position to lOffset from the end of the data stream.

  - IOSEEK_CURRENT: Move the data stream position to lOffset from the current position in the data stream.

- lOffset: The number of bytes to move the data stream position. May be positive or negative.

#### Return Values

- IOERR_OK: The current stream position was successfully changed.
- IOERR_UNKNOWN: The current stream position was not changed.

## IOTell

Reports the current data stream position.

```
IOERR IOTell(BASEIO* pBaseIO, VTLPDWORD pOffset);
```

#### Parameters

- pBaseIO: A pointer to the BASEIO structure for this data stream.
- pOffset: Pointer to a DWORD that should be set to the current data stream position.

#### Return Value

- IOERR_OK: Current position was reported.
- IOERR_UNKNOWN: Failure to find or report the current position.

## IOGetInfo

This is a multi-purpose function that is used for querying the IO Provider for information about the data stream and how it should be referenced in transformation output. Some of the queries may be ignored, while others are required to be handled.

The IOGetInfo function has the following prototype:

```
IOERR IOGetInfo(BASEIO* pBaseIO, VTDWORD dwInfoId, VTLPVOID
   pInfo);
```

#### Parameters

- pBaseIO: A pointer to the BASEIO structure for this data stream.
- dwInfoId: The Info ID of the info request being made.
- pInfo: Pointer to auxiliary data that some info requests pass, or require to be returned.

#### Return Value

- IOERR_OK: The requested information was supplied.
- IOERR_BADINFOID: The specified query is not supported.
- IOERR_UNKNOWN: The requested information is not available.
- IOERR_FALSE: Returned in response to specific queries documented below.

## IOGetInfo Info IDs

The following ID values are defined for dwInfoId. Pay particular attention to notes indicating that a value is required in order for transformations to succeed.

### IOGETINFO_FILENAME_IOP

This message retrieves the name of the data stream. If the data stream represents a file in some non-file-system based repository, the file name is what should be returned from this message.

pInfo points to a TS_stringData structure. The IO Provider must allocate the str field within this structure using the Alloc function supplied in the IO Consumer interface. The name is to be copied into this allocated structure. The IO Consumer will be responsible for freeing the allocated memory.

This query must always be handled.

Example:

```
TS_IOSpec * pFilename = (TS_IOSpec *) pInfo;
MyIOStruct * pMyData = (MyIOStruct *) pBaseIO;
pInfo->str = pMyData->pConsumer->Alloc( strlen(pMyData->name)+1 );
strcpy( pInfo->str, pMyData->name );
```

### IOGETINFO_PATHNAME_IOP

This message retrieves the full path to the data stream. If the data stream represents a file in some non-file-system based repository, and has path information associated with it, the path to the file is what should be returned from this message. If there is no appropriate response to this message, IOGetInfo should return IOERR_UNKNOWN.

pInfo points to a TS_stringData structure. The IO Provider must allocate the str field within this structure using the Alloc function supplied in the IO Consumer interface. The path is to be copied into this allocated structure. The IO Consumer will be responsible for freeing the allocated memory.

This query must always be handled.

Example:

```
TS_stringData * pFilename = (TS_stringData *) pInfo;
MyIOStruct * pMyData = (MyIOStruct *) pBaseIO;
pInfo->str = pMyData->pConsumer->Alloc( strlen(pMyData->path)+1 );
strcpy( pInfo->str, pMyData->path );
```

### IOGETINFO_HYPERLINK - HTML Export Only

This message retrieves the URL of the data stream, and is used to generate links between transformation output documents.

pInfo points to a TS_stringData structure. The IO Provider must allocate the str field within this structure using the Alloc function supplied in the IO Consumer interface. The URL is to be copied into this allocated structure. The IO Consumer will be responsible for freeing the allocated memory.

This query must be handled when the output of the transformation is handled via the IO provider. If the IO provider is supporting only the input side of the transformation, this query will not be received.

This query must be handled when the output of the transformation is handled via the IO provider. If the IO provider is supporting only the input side of the transformation, this query will not be received.

Example:

```
TS_stringData * pFilename = (TS_stringData *) pInfo;
MyIOStruct * pMyData = (MyIOStruct *) pBaseIO;
pInfo->str = pMyData->pConsumer->Alloc( strlen(pMyData->url)+1 );
strcpy( pInfo->str, pMyData->url );
```

#### IOGETINFO_GENSECONDARY_IOP

This message is sent by an IO Consumer that needs to open an additional data stream for reading, which may occur with certain types of input documents that refer to other documents, or when transformation templates refer to secondary templates.

This query should be handled to support the complete set of input formats supported by Outside In, and the full range of template functionality. If not this query is not handled, transformations that use templates that refer to other templates will fail, as will transformations of some types of input documents.

pInfo points to an IOGENSECONDARY_IOP structure:

```
typedef struct IOGENSECONDARY_IOP
{
   TS_stringData   filename;
   TS_IOSpec       ioSpec;
   VTDWORD         dwOpenFlags
} IOGENSECONDARY_IOP, * PIOGENSECONDARY_IOP;
```

- filename: Set by the caller. This is the name of the secondary data stream to be opened. In most cases there will be no "path" information. To use the analogy of files in a file system, the location of the current data stream would be the "current directory", and the data stream indicated by the filename parameter would be assumed to exist in the same "directory."

- ioSpec: To be filled in by the IO provider in response to this message. An IO specification for the new document that may be used in a subsequent call to OpenIO to open the data stream. The IO Provider must allocate the spec.str and "specType" fields within this structure using the Alloc function supplied in the IO Consumer interface. The caller will be responsible for freeing these strings.

- dwOpenFlags: A set of flags indicating how the secondary file should be opened. Multiple flags may be use by bitwise OR-ing them together. One of the following values:

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

  - IOOPEN_WRITE: The secondary file should be opened for write.

  - IOOPEN_CREATE: The secondary file should be created and then opened.

#### IOGETINFO_CREATENEWIOSPEC

This message is sent by an IO Consumer when it needs to create an additional output data stream. During a transformation, the BASEIO associated with the primary output sink will receive this message for all additional output streams that need to be created. This message provides hints for the name and format of the output data stream, and requires that the IO Provider return an IO specification that can be used in a subsequent call to OpenIO.

This query must be handled for an IO Provider to support creation of output documents. It is not used for input documents.

pInfo points to an IOCREATENEWIOSPEC structure:

```
struct IOCREATENEWIOSPEC
{
   IOSpec   ioSpec;
   VTCHAR   *suggestedName;
   VTCHAR   *outputType;
} IOCREATENEWIOSPEC, * PIOCREATENEWIOSPEC;
```

- ioSpec: To be filled in by the IO provider in response to this message. A specification for a new document that is going to be created by the transformation process. The IO Provider must allocate the spec.str and specType fields within this structure using the Alloc function supplied in the IO Consumer interface. The caller will be responsible for freeing these strings.
- suggestedName: Set by the caller. A string, in UTF-8 encoding, that provides a suggested name for this output document. This string may be interpreted as the "base name" for the output document. (In a file system, this would be the portion of the file name prior to the extension.) This string pointer may also be null, indicating that there is no suggestion for the document name.
- outputType: Set by the caller. A string, in UTF-8 encoding, that identifies the type of document for which the new specification will apply. This string may be interpreted as the default "file extension" for the new output document. The values for this string may include html, xml, gif, jpg, png, or another extension that may have been defined for other supported output formats. This string pointer may also be null, indicating that the file type is unknown.

#### IOGETINFO_PROVIDERDATA

This method is provided for the private use of the implementer. It is provided so that a developer who implements both a Transformation Engine and an IO Provider would have a convenient way to establish a private communication between the two.

This method is not supported when the IO Provider is executing on the client side.

This query is optional.

# IO Consumer Interface

The IO Consumer interface is a set of functions provided and implemented by Transformation Server to be used by an IO Provider for various activities related to opening, reading and writing sources of input and output.

Like the BASEIO interface, the IO Consumer interface consists of a data structure (IOConsumerInterface) containing pointers to functions. The pointer to this data structure must be handed back to each of the IO Consumer functions.

## Alloc

Certain operations in the BASEIO interface require the IO Provider to allocate memory, to be freed by the IO Consumer. The IO Consumer's Alloc function must be used for these allocations.

### Prototype

```
typedef VTLPVOID (IO_CALLTYPE* IOCAllocProc)(const struct
   IOConsumerInterface* pConsumer, VTDWORD dwSize);
```

### Parameter

- dwSize: The size, in bytes, of the memory to be allocated.

### Return Values

Returns either a valid, non-null pointer to memory, or if the allocation fails, a null pointer.

## Free

This function frees memory allocated via the IO Consumer's Alloc function. Note that this function should never be called to free memory that was returned to the IO Consumer via IOGetInfo operations.

This function is provided to allow the IO Provider to use the Alloc function for private memory allocations. Use of this function is strictly optional; when an IO source is closed, all of the memory allocated via the IO Consumer's Alloc function will be automatically cleaned up.

### Prototype

```
typedef VTVOID (IO_CALLTYPE* IOCFreeProc)(const struct
   IOConsumerInterface* pConsumer, VTLPVOID pMem);


void Free(VTLPVOID pMem)
```

### Parameters

- pMem: A pointer to memory to be deallocated. The memory being deallocated must have been allocated using Alloc( ).

### Return Values

none

## UTF8toUCS2

Converts a UTF-8 string to UCS2 (wide character) using an algorithm from the Unicode Standard 2.0.

### Prototype

```
typedef TSERR (IO_CALLTYPE* UTF8toUCS2Proc) (const struct
   IOConsumerInterface* pConsumer, unsigned char*  sourceStart,
   wchar_t* targetStart, VTDWORD* pTargetSize);
```

### Parameters

- sourceStart: The input UTF-8 character string. Must be null-terminated.
- targetStart: The wide character array which will contain the Unicode string.
- pTargetSize: Pointer to the variable that contains the size of the buffer for the UCS-2 string. Upon return, this will be set to the size in wide characters (not bytes) of the decoded string, including the terminating null. If this function returns TSERR_BOUNDSEXCEEDED, the pTargetSize will still be set to the full size of the correctly decoded string and can be used to determine an appropriate buffer size for a second attempt to decode the string.

### Return Values

- TSERR_OK: The conversion succeeded.
- TSERR_BADPARAM: The sourceStart or pTargetSize is null.
- TSERR_BOUNDSEXCEEDED: The UTF-8 string buffer wasn't large enough.

## UCS2toUTF8

Converts a UCS2 string to UTF-8 string using an algorithm from the Unicode Standard 2.0.

### Prototype

```
typedef TSERR (IO_CALLTYPE* UCS2toUTF8Proc) (const struct
   IOConsumerInterface* pConsumer, wchar_t* sourceStart,
   unsigned char* targetStart, VTDWORD* pTargetSize);
```

### Parameters

- sourceStart: The input UCS2 character string. Must be null-terminated.
- targetStart: The byte array which will contain the UTF-8 string.
- pTargetSize: Pointer to the variable that contains the size of the buffer for the UTF-8 string. Upon return, this will be set to the size (in bytes) of the decoded string, including the terminating null. If this function returns TSERR_BOUNDSEXCEEDED, the pTargetSize will still be set to the full size of the correctly decoded string and can be used to determine an appropriate buffer size for a second attempt to decode the string.

### Return Values

- TSERR_OK: The conversion succeeded.
- TSERR_BADPARAM: The sourceStart or pTargetSize is null.
- TSERR_BOUNDSEXCEEDED: The UTF-8 string buffers wasn't large enough.

## IOConsumerInterface Data Structure

The IOConsumerInterface structure has the following definition:

```
typedef VTLPVOID (ENTRYMOD* IOCAllocProc)( const struct
   IOConsumerInterface* pConsumer, VTDWORD size);
typedef VTVOID (ENTRYMOD* IOCFreeProc)( const struct
   IOConsumerInterface* pConsumer, VTLPVOID pMem);
typedef IO_ENTRYPOINT TSERR (IO_CALLTYPE* UTF8toUCS2Proc)
   (const struct IOConsumerInterface* pConsumer,
   unsigned char*   sourceStart,
   wchar_t*         targetStart,
   VTDWORD*         targetSize);
typedef IO_ENTRYPOINT  TSERR (IO_CALLTYPE* UCS2toUTF8Proc)
   (const struct IOConsumerInterface* pConsumer,
   wchar_t*         sourceStart,
   unsigned char*   targetStart,
   VTDWORD*         pTargetSize);

typedef struct IOConsumerInterface
{
   VTDWORD          version;
```

```
   IOCAllocProc     Alloc;
   IOCFreeProc      Free;
   UTF8toUCS2Proc   UTF8toUCS2;
   UCS2toUTF8Proc   UCS2toUTF8;
} IOConsumerInterface,  * PIOConsumerInterface;
```

- version: This specifies the version number of the transformation agent API specification to which this agent was written. The format of this number is not currently documented, but later versions are guaranteed to have a higher version number than earlier versions. Developers should compare this value to kIOConsumerInterfaceVersion, which is the current version of the IO Consumer interface. If the IO Provider does not support the specified version, it should set the version field to a version number that it does support and return IOERR_FALSE from the OpenIO function.
- Alloc: Points to the agent's IO Consumer's Alloc function.
- Free: Points to the agent's IO Consumer's Free function.
- UTF8toUCS2: Provides a function to transform characters from 8-bit UTF-8 encoding to 16-bit Unicode UCS-2 encoding.
- UCS2toUTF8: Provides a function to transform characters from 16-bit Unicode UCS-2 encoding to Unicode 8-bit UTF-8 encoding.

# 8

# Upgrading Applications to Use Transformation Server

This chapter is a guide to incorporating Transformation Server, using its C language client module (SCCTS) into applications that already use the embedded version of the Outside In Export technologies (the SCCEX and SCCDA modules).

This chapter assumes that the reader is familiar with the Outside In API.

This chapter includes the following sections:



## Basic Transformation Operations

These are the basic tasks involved in updating an application from the embedded Outside In Export APIs to the Transformation Server APIs:

1. Configure the application to link with SCCTS instead of SCCEX and sccda.
2. Replace calls to DAInit and DADeinit with calls to TSInit and TSDeinit.
3. Most calls to DASetOption can be updated by calling TSSetOptionById with identical parameters. Some exceptions to this rule (including DASetFileSpecOption) will be documented below.
4. Replace callback function with additional calls to TSSetOption or TSSetOptionById.
5. Replace call to EXRunExport with call to TSRunTransform.

## Initialization and De-initialization

The functions DAInit and DADeinit can just be replaced with calls to TSInit and TSDeinit.

TSInit should be called upon loading SCCTS. It needs to be called only once per process. However, in situations where TSInit is called multiple times, only the first call causes it to initialize. Subsequent calls are ignored, but are counted by an internal reference counter. Therefore, each call to TSInit must be balanced by a corresponding call to TSDeinit. The last call to TSDeInit causes sccts to actually de-initialize.

Unlike DAInit, TSInit requires some parameters be provided. These parameters specify where Transformation Server can be found.

### Embedded Version

```
if( DAERR_OK == DAInit() )
    ; /* everything is OK */
```

### C Client Version

```
/* Assuming transformation server is listening for requests
   on port 999 of the local host (IP address 127.0.0.1) */
TSINITPARAMS2 tsinit = {0};
tsinit.dwVersion    = SCCTS_INITPARAMS_CURRENTVERSION;
tsinit.szServer     = "127.0.0.1"; /* hostname */
                                   IP address of where client side
                                   redirected IO is taking place,
                                   needed only when source or sink uses
                                   redirected IO. (null terminated) */
init.wPort        = 999;
tsinit.openIO       = NULL;        /* points to redirected IO
                                   /* open */
tsinit.wIOPort      = 0;           /* function used when */
                                   /* providing redirected IO */

if( TSERR_OK = TSInit(&tsinit) )
    ;  /* everything is OK */
```

TSDeinit should be called immediately prior to unloading SCCTS, or upon application exit. Like DADeinit, it requires no parameters. It can be simply swapped for existing calls to DADeinit.

# Setting Transformation Parameters

This section describes transformation parameters.

## Options

The following information concerns options.

### Replacing the Document Handle with an "Option Set Handle"

In the embedded versions of the Outside In Export interfaces, options are tied to a "document handle" – an identifier returned from DAOpenDocument that tells the Export module to which input document the options should apply. The Transformation Server C client presents a different model: the input document is not "opened" in any way prior to initiating a transformation; and transformation options are grouped together in an "option set" that may be applied to more than one input document.

### Embedded Version

```
VTHDOC hDoc;     /* document handle */
DAERR deResult ; /* result code*/
deResult = DAOpenDocument(
    &hDoc,                      /* receives handle to document */
    IOTYPE_ANSIPATH,            /* type of path to file */
    (VTLPVOID) pInputPath,      /* input file */
    0);                         /* flags */
/* when the document no longer needs to be open */
DACloseDocument(hDoc);
```

### C Client Version

```
TSOPTIONS hOpt;     /* option set handle */
TSERR     tsResult; /* result code */
tsResult = TSOpenOptions (
    NULL,    /* name of option set (null term. string or NULL)*/
    &hOpt ); /* receives handle to option set */
/* when the options are no longer needed */
TSCloseOptions(hOpt);
```

## Setting Options

The SCCTS module includes a function called TSSetOptionById that supports most of the data structures and option identifiers that were used in the embedded interface function DASetOption. Apart from their names, the functions differ only in the first parameter, which controls how the options are collected (with a "document handle" vs. an "option set handle").

Most options that affect a transformation can be specified with the same data types that were used in the interface to SCCEX. For these options, simply replace the VTHDOC parameter with a TSOPTIONS parameter, and call TSSetOptionById instead of DASetOption. The following is an example:

### Embedded Version

```
DAERR deResult; /* result code*/
DWORD dwVal = FI_GIF;
    deResult = DASetOption( hDoc, SCCOPT_GRAPHIC_TYPE,
        (LPVOID)&dwVal, sizeof(DWORD));
```

### C Client Version

```
TSERR tsResult; /* result code */
DWORD dwVal = FI_GIF;
    tsResult = TSSetOptionById( hOpt, SCCOPT_GRAPHIC_TYPE,
        (LPVOID)&dwVal, sizeof(DWORD));
```

## Exceptions to This Rule (HTML Export Only)

The option SCCOPT_EX_TEMPLATE, which identifies the template to be used when producing output documents, must be specified with a different data type than it was in the embedded API.

In the embedded API this option was specified via a special function called DASetFileSpecOption. In the C client API the template location is described in a TS_IOSpec data structure, which is specified through the TSSetOptionById function. As a result of this difference, the C client uses a different identifier, defined as SCCOPT_TS_TEMPLATE, that is specific to the C client. Any attempt to specify an option identified as SCCOPT_EX_TEMPLATE will fail.

### Embedded Version

```
deResult = DASetFileSpecOption( hDoc, SCCOPT_EX_TEMPLATE,
    IOTYPE_ANSIPATH, "\\exports\templates\template.html" );
```

### C Client Version

```
TS_IOSpec template;
    template.spec.str = "\\exports\templates\template.html";
    template.spec.charset = ts_windows_1252;
    template.specType = "path";
    tsResult = TSSetOptionById( hOpt, SCCOPT_TS_TEMPLATE,
        (LPVOID)&template, sizeof(TS_IOSpec));
```

## Callbacks

The caller-supplied callback function that was defined in the embedded API does not exist for the Transformation Server C Client API. Where possible, the callback messages have been replaced by new options with equivalent functionality. In other cases, the functionality represented by callback messages is supported in a limited form or not supported by the C Client API. The following sections offer explanations for how each embedded API callback message is supported in the C Client API.

### EX_CALLBACK_ID_CREATENEWFILE

This functionality represented by this callback has been distributed to different areas of the client API. This callback existed to notify the calling application when a new output file was about to be created, allowing the application to change the both the name and the hyperlink (URL) used to reference the file in the transformation output.

For an application using Transformation Server's built-in input and output facilities, the ability to change the names of the output file or its URL will not be supported in the Transformation Server client. For applications that provide their own IO through the redirected IO interface, this callback's functionality will be supported through the IOGetInfo messages IOGETINFO_CREATENEWOUTPUTSPEC (and IOGETINFO_HYPERLINK for HTML Export). For details, please refer to the documentation for redirected IO.

The reporting capabilities of this message are provided by the results of the TSRunTransform function, which provides an array containing the specifications of all output files created for the transformation.

### EX_CALLBACK_ID_NEWFILEINFO

The reporting capabilities of this message are provided by the results of the TSRunTransform function, which provides an array containing the specifications of all output files created for the transformation.

### EX_CALLBACK_ID_ALTLINK (HTML Export Only)

This callback's functionality has been replaced by a new option: SCCOPT_TS_ALTLINK. This option is specified as a data structure (OIT_AltLink) containing two TS_stringData structures, one for the "previous" and one for the "next" alt links.

#### Embedded Version

```
case EX_CALLBACK_ID_ALTLINK:
{
   EXALTLINKCALLBACKDATA *pData =
      (EXALTLINKCALLBACKDATA*)pCommandOrInfoData;
   result = SCCERR_NOTHANDLED;

   if( pData->dwType == EX_ALTLINK_PREV )
   {
      lstrcpy(pData->pAltURLStr, "alternate prev-page link");
      result = SCCERR_OK;
   }
   else if( pData->dwType == EX_ALTLINK_NEXT )
   {
      lstrcpy(pData->pAltURLStr, "alternate next-page link");
      result = SCCERR_OK;
   }
}
break;
```

#### C Client Version

```
HWX_AltLink   altLinks;
altLinks.prev.str = "alternate prev-page link";
altLinks.prev.charset = ts_windows_1252;
altLinks.next.str = "alternate next-page link";
altLinks.next.charset = ts_windows_1252;

TSSetOptionById( hOpt, SCCOPT_TS_ALTLINK, (VTLPVOID)&altLinks,
   sizeof(HWX_AltLink);
```

### EX_CALLBACK_ID_PROCESSLINK (HTML Export Only)

This functionality represented by this callback message has limited support in Transformation Server. While the option to insert alternate links is not available, the basic ability to skip or process linked graphics is presented as a Boolean option with the identifier SCCOPT_TS_SKIPLINKEDIMAGES. This option allows the calling application to specify that linked files should be either transformed or skipped.

#### C Client Example

```
VTBOOL   bSkipLinkedImages = TRUE;
TSSetOptionById( hOpt, SCCOPT_TS_SKIPLINKEDIMAGES,
   &bSkipLinkedImages, sizeof(VTBOOL) );
```

### Unsupported Callbacks

The functionality represented by the following embedded API callback messages is not supported on the Transformation Server client side:

- EX_CALLBACK_ID_CUSTOMELEMENTLIST
- EX_CALLBACK_ID_GRAPHICEXPORTFAILURE
- EX_CALLBACK_ID_OEMOUTPUT
- EX_CALLBACK_ID_OEMOUTPUT_VER2
- EX_CALLBACK_ID_PROCESSELEMENTSTR
- EX_CALLBACK_ID_PROCESSELEMENTSTR_VER2
- EX_CALLBACK_ID_REFLINK
- EX_CALLBACK_ID_ENTERARCHIVE
- EX_CALLBACK_ID_LEAVEARCHIVE

# Performing a Transformation

This information pertains to performing transformations.

## Specifying Inputs and Outputs with TS_IOSpec

The input and output documents for a transformation (also referred to as the "source" and "sink", respectively) are specified by a new data structure named TS_IOSpec.

```
typedef struct TS_IOSpec
{
   TS_stringData spec;    /* specifies the "path" of the doc */
   XSD_string specType;   /* specifies the type of path being
                     /* specified */
} TS_IOSpec;
```

The specification of the IO target itself is described by another new data structure, TS_stringData, which has this definition:

```
typedef struct TS_stringData
{
   TS_char* str;                     /* contents of string */
   enum TS_CharacterSetEnum charset; /* char. set of string */
} TS_stringData;
```

The specType field of TS_IOSpec is analogous to the embedded API's IOTYPE identifiers, but is stricter in its definition: it describes whether the specification is of a file-system path, a URL, a redirected IO type, or some custom IO type. It does *not* imply anything about how that specification is encoded.

### Examples

```
TS_IOSpec   input;   /* specifying a file path */
   input.spec.str = "c:\documents\important stuff.doc";
   input.spec.charset = ts_windows_1252;
   input.specType = "path";

TS_IOSpec output;    /* specifying a url */
   output.spec.str =
      "http://intranet.company.com/docs/important.htm";
   output.spec.charset = ts_windows_1252;
   output.specType = "url";

TS_IOSpec moreInput; /* specifying a redirected IO source */
   moreInput.spec.str = "app.customDATABASE:r244.field8";
   moreInput.spec.charset = ts_windows_1252;
   moreInput.spec.specType = "redirect";
```

## Initiating the Transformation

Once all of the transformation options have been specified, the calling application may now trigger a transformation operation. This is done by replacing the call to EXRunExport with a call to TSRunTransform.

### Embedded Version

```
deResult = EXRunExport( hDoc, ... )
```

### C Client Version

```
TS_IOSpec input, output;
TSERR tsResultCode;
TSOPTIONS hOpt;
TS_TransformResult * pResults;
/* ... initialize options, input and output specs ... */
   tsResultCode = TSRunTransform(
   &input,    /* source document */
   &output,   /* sink (output) document */
   "html",    /* desired output format */
   NULL,      /* named option set on server (may be NULL) */
   hOpt,      /* option set handle (may be NULL) */
   &pResults  /* data describing results of transformation */
   );
```

It should also be noted that the data returned in the TS_TransformResult pointer must be freed by the calling application. This is done through a single call to TSMemFree:

```
if( pResults )
{
   /* make use of the results, then dispose of them... */
```

```
   TSMemFree(pResults);
}
```

## Inspecting the Results

In addition to the numeric error code returned from TSRunTransform, a data structure, TS_TransformResult, is also filled in that provides a human-readable result message and a list of the output documents created by the transformation. The TS_TransformResult data structure has the following definition:

```
typedef struct TS_TransformResult
{
    TSERR          resultCode;   /* numeric error code, same as
                                 return value from
                                 TSRunTransform */
    TS_stringData  resultString; /* descriptive text for result;
                                 may be empty. */
    TS_OutputList  outputList;   /* contains list of IO specs
                                 for transformation output; may
                                 be empty if error occurred. */
} TS_TransformResult;
```

This data structure makes use of another data structure, TS_OutputList:

```
typedef struct TS_OutputList
{
   TS_IOSpec *  documents;        /* array of output document
                                  specifications */
   XSD_unsignedInt size;       /* number of items in the array */
} TS_OutputList;
```

# Advanced Transformation Operations

This section pertains to advanced transformation options.

## Handling Redirected IO

Like the embedded version of the Outside In Export API, Transformation Server supports extension of its IO facilities, through a method we call "redirected IO." If your application has implemented redirected IO, you'll find that most of your code will not have to be changed, though it may need to be reorganized a little bit.

### Server-Side vs. Client-Side Redirected IO

As a client-server application, Transformation Server runs its transformation operations in a separate process from the "client" application that uses it. For maximum flexibility, Transformation Server allows redirected IO to be provided on either the server side or the client side. In other words, the code that provides the IO may execute in the process of the client application or in the process where the transformation occurs. From an implementation point of view, there is little to no difference between the two approaches; it is entirely possible to use the same binary code to provide redirected IO on both the client and server.

As in the embedded API, an application that provides redirected IO must implement the functions defined in the BASEIO interface. In client-side redirected IO, the C client API communicates through TCP/IP with the server-side transformation process, and relays IO operations to the BASEIO interface provided by the application. In server-side redirected IO, the application provides its redirected IO code in a dynamically loadable library that is loaded by the transformation process as needed. Communication with the redirected IO code then proceeds in-process as the transformation is being performed.

While the server-side redirected IO may require a small amount of additional work when compared to client-side redirected IO, it has a significant performance advantage in the fact that all IO occurs in-process. Client-side IO, on the other hand, has the ability to communicate in-process with the client application that is requesting the transformation, which may be a requirement for retrieving or writing the data.

## What's Different About Redirected IO in Transformation Server

From an implementation point of view, the main differences between redirected IO in Transformation Server versus the embedded API are:

1. How the IO "targets" are specified in the API.
2. How IOGetInfo queries are handled.

### Specifying IO Targets

In the embedded API, the application provides the Outside In Export module with a pointer to a BASEIO structure that contains pointers to all of the IO functions that will be used to access one specific target. Transformation Server, however, requires that applications specify IO targets with an IO specification in the same manner that file-system documents are specified. The application must then also provide an "open" function that is used by Transformation Server to obtain a BASEIO structure from the IO specification.

Here's how IO targets are specified using the embedded API:

```
struct myIOStruct
{
   BASEIO   baseIO;
   MYDATA   privateData;
};
/* ... initialize myIOStruct ...*/
EXRunExport( hDoc, (HIOFILE) &myIOStruct, FI_HTML, ... )
```

In the server API, specifying an IO target is a two-step process:

1. Specifying an IO target:

```
TS_IOSpec   input;
input.spec.str = "my.private.IO.specification";
input.spec.charset = ts_UTF_8;
input.specType = "redirect";  /* this value is required for */
                              /* specType on the client side */
```

2. Resolving the IO target:

```
TSERR OpenIO( TS_IOSpec *pSpec, DWORD dwFlags,
   IOConsumerInterface *pConsumer, BASEIO *pIOResults )
{
   struct myIOStruct theIOStruct;
   /* inspect contents of pSpec to determine your IO target */
   /* inspect dwFlags for read/write/create options */
   /* initialize theIOStruct */
   *pIOResults = theDoc;
   return TSERR_OK;
}
```

This approach to redirected IO is what allows the code that provides redirected IO to execute on the server side as well as the client.

### IOGetInfo Messages

The IOGetInfo function has defined some new messages and redefined some existing ones, to accommodate the differences between the embedded architecture and the

Transformation Server architecture. The Redirected IO portion of this manual gives the detailed description of what is required of your IOGetInfo function, but some of the chief differences are:

1. The use of the callback query SCCEX_CALLBACK_CREATENEWFILE has been replaced with IOGetInfo the queryies IOGETINFO_CREATENEWIOSPEC (and IOGETINFO_HYPERLINK for HTML Export).
2. The IOGetInfo queries IOGETINFO_PATHNAME, IOGETINFO_FILENAME, and IOGETINFO_GENSECONDARY_IOP have been replaced with new versions that use the Transformation Server's TS_stringData structure to provide unambiguous string and character set information.

### Redirected IO on the Client Side

The following steps are necessary to take an existing implementation of redirected IO and make it available to the Transformation Server C Client API.

1. Define a way to represent an IO target in a text string that can be passed in a TS_IOSpec structure.
2. Implement an OpenIO function that maps from TS_IOSpec to your BASEIO structure.
3. Modify your IOGetInfo function to handle the new Transformation Server style queries.
4. Provide the pointer to your OpenIO function in the initialization structure passed to TSInit.

### Redirected IO on the Server Side

To make redirected IO available on the server side, perform steps 1 through 3 above, and then do the following:

1. Build your redirected IO code into a loadable module/DLL, with OpenIO specified as an exported function.
2. Modify the Transformation Server configuration file agent_iospec_types.xml to indicate the location of your IO module.

In the preceding step, you must also define an IO spec type for your module (the specType field in the TS_IOSpec structure). Note that the specType value redirect is reserved for client-side redirected IO, and will not work for a server-side IO provider.

Note that these two approaches are not mutually exclusive. There's no reason an IO Provider built for server-side redirected IO couldn't be used on the client side, as well.

## How Embedded API Options Map to the New SOAP Options

The following table shows how the option names from the embedded technology map to the new option names used by the SOAP interface in Transformation Server. Embedded API options which do not have a corresponding option in the SOAP API are marked as "NA".

For details about the options discussed here, see the Options documentation for your system.

## XML Export

This information pertains to XML Export.

| Embedded API | SOAP API |
|---|---|
| SCCOPT_ACCEPT_ALT_GRAPHICS | acceptAlternateGraphics |
| SCCOPT_CCFLEX_FORMATOPTIONS | Each of the flags in the embedded option has a matching, stand-alone Boolean option in the SOAP API:<br>charMappingBoth<br>charMappingText<br>charMappingNone<br>charMappingDefault<br>convertChartObjects<br>convertDateTimeProperties<br>convertImageObjects<br>convertPresentationObjects<br>convertVectorObjects<br>delimiters<br>flattenStyles<br>generateSystemData<br>noBitmapElements<br>noChartElements<br>noPresentationElements<br>noVectorElements<br>separateStyleTables<br>useFullFilePath |
| SCCOPT_CCFLEX_INCLUDETEXTOFFSETS | includeTextOffsets |
| SCCOPT_CCFLEX_REMOVEFONTGROUPS | removeFontGroups |
| SCCOPT_DEFAULTINPUTCHARSET | defaultInputCharset |
| SCCOPT_EX_CALLBACKS | NA |
| SCCOPT_EX_UNICODECALLBACKSTR | NA |
| SCCOPT_EXXML_DEF_METHOD | xmlDefinitionMethod |
| SCCOPT_EXXML_DEF_REFERENCE | xmlDefinitionLocation |
| SCCOPT_EXXML_SUBSTREAMROOTS | subStreamRoots |
| SCCOPT_FALLBACKFORMAT | fallbackFormat |
| SCCOPT_FIFLAGS | extendedTestForText |
| SCCOPT_FILTERJPG | allowJPEG |
| SCCOPT_FILTERLZW | allowLZW |
| SCCOPT_FORMATFLAGS | isoDateTimes |
| SCCOPT_GIF_INTERLACED | graphicGifInterlaced |
| SCCOPT_GRAPHIC_HEIGHTLIMIT | graphicHeightLimit |
| SCCOPT_GRAPHIC_OUTPUTDPI | graphicOutputDPI |
| SCCOPT_GRAPHIC_SIZELIMIT | graphicSizeLimit |

| Embedded API | SOAP API |
|---|---|
| SCCOPT_GRAPHIC_SIZEMETHOD | graphicSizeMethod |
| SCCOPT_GRAPHIC_TYPE | graphicType |
| SCCOPT_GRAPHIC_WIDTHLIMIT | graphicWidthLimit |
| SCCOPT_IO_BUFFERSIZE | Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API:<br>readBufferSize<br>memoryMappedInputSize<br>tempBufferSize |
| SCCOPT_JPEG_QUALITY | graphicJpegQuality |
| SCCOPT_RENDERING_PREFER_OIT | preferOITRendering |
| SCCOPT_REORDERMETHOD | reorderMethod |
| SCCOPT_TEMPDIR | NA |
| SCCOPT_TIMEZONE | timezone |
| SCCOPT_UNMAPPABLECHAR | unmappableCharacter |
| SCCOPT_XML_DEF_METHOD | xmlDefinitionMethod |
| SCCOPT_XML_DEF_REFERENCE | xmlDefinitionLocation |

## PDF Export

This information pertains to PDF Export.

| Embedded API | SOAP API |
|---|---|
| SCCOPT_APPLYFILTER | applyZLIB |
| SCCOPT_DBPRINTFITTOPAGE | databaseFitToPage |
| SCCOPT_DBPRINTGRIDLINES | databaseShowGridLines |
| SCCOPT_DBPRINTHEADINGS | databaseShowHeadings |
| SCCOPT_DEFAULTINPUTCHARSET | defaultInputCharset |
| SCCOPT_DEFAULTPRINTFONT | defaultFont |
| SCCOPT_DEFAULTPRINTMARGINS | defaultMargins |
| SCCOPT_ENABLEWATERMARK | enableWatermark |
| SCCOPT_EX_CALLBACKS | NA |
| SCCOPT_DEFAULTPAGESIZE | The functionality of this option is supported by three options in the server implementation:<br>defaultPageUnits<br>defaultPageHeight<br>defaultPageWidth |
| SCCOPT_DOLINEARIZATION | doLinearization |
| SCCOPT_EMBEDFONTS | embedFonts |
| SCCOPT_EX_UNICODECALLBACKSTR | NA |

| Embedded API | SOAP API |
|---|---|
| SCCOPT_FALLBACKFORMAT | fallbackFormat |
| SCCOPT_FIFLAGS | extendedTestForText |
| SCCOPT_FILTERJPG | allowJPEG |
| SCCOPT_FILTERLZW | allowLZW |
| SCCOPT_FONTDIRECTORY | fontDirectory |
| SCCOPT_FONTFILTER | The functionality of this option is handled by two options in the server implementation:<br>excludeFont<br>includeFont |
| SCCOPT_FORMATFLAGS | isoDateTimes |
| SCCOPT_GRAPHIC_OUTPUTDPI | graphicOutputDPI |
| SCCOPT_GRAPHIC_SIZEMETHOD | graphicSizeMethod |
| SCCOPT_IO_BUFFERSIZE | Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API:<br>readBufferSize<br>memoryMappedINputSize<br>tempBufferSize |
| SCCOPT_MAXSSDBPAGEHEIGHT | maxSsDbPageHeight |
| SCCOPT_MAXSSDBPAGEWIDTH | maxSsDbPageWidth |
| SCCOPT_PRINTENDPAGE | endPage |
| SCCOPT_PRINTFONTALIAS | fontAlias |
| SCCOPT_PRINTSTARTPAGE | startPage |
| SCCOPT_RENDERING_PREFER_OIT | preferOITRendering |
| SCCOPT_REORDERMETHOD | reorderMethod |
| SCCOPT_SSPRINTDIRECTION | spreadsheetPageDirection |
| SCCOPT_SSPRINTFITTOPAGE | spreadsheetFitToPage |
| SCCOPT_SSPRINTGRIDLINES | spreadsheetShowGridLines |
| SCCOPT_SSPRINTHEADINGS | spreadsheetShowHeadings |
| SCCOPT_SSPRINTSCALEPERCENT | spreadsheetScalePercentage |
| SCCOPT_SSPRINTSCALEXHIGH | spreadsheetScaleXPagesHigh |
| SCCOPT_SSPRINTSCALEXWIDE | spreadsheetScaleXPagesWide |
| SCCOPT_TEMPDIR | NA |
| SCCOPT_TIMEZONE | timezone |
| SCCOPT_UNMAPPABLECHAR | unmappableCharacter |
| SCCOPT_USEDOCPAGESETTINGS | useDocumentPageSettings |
| SCCOPT_WATERMARKIO | The functionality of this option is supported by three options in the server implementation: |

| Embedded API | SOAP API |
|---|---|
| | watermarkImage<br>watermarkScaling<br>watermarkScalePercent |
| SCCOPT_WATERMARKPOSITION | watermarkPosition |
| SCCOPT_WHATTOPRINT | usePageRange |

## Image Export

This information pertains to Image Export.

| Embedded API | SOAP API |
|---|---|
| SCCOPT_DBPRINTFITTOPAGE | databaseFitToPage |
| SCCOPT_DBPRINTGRIDLINES | databaseShowGridLines |
| SCCOPT_DBPRINTHEADINGS | databaseShowHeadings |
| SCCOPT_DEFAULTINPUTCHARSET | defaultInputCharset |
| SCCOPT_DEFAULTPRINTFONT | defaultFont |
| SCCOPT_DEFAULTPRINTMARGINS | defaultMargins |
| SCCOPT_EX_CALLBACKS | NA |
| SCCOPT_EX_UNICODECALLBACKSTR | NA |
| SCCOPT_FALLBACKFORMAT | fallbackFormat |
| SCCOPT_FIFLAGS | extendedTestForText |
| SCCOPT_FILTERJPG | allowJPEG |
| SCCOPT_FILTERLZW | allowLZW |
| SCCOPT_FORMATFLAGS | isoDateTimes |
| SCCOPT_GIF_INTERLACED | graphicGifInterlaced |
| SCCOPT_GRAPHIC_CROPPING | graphicCropping |
| SCCOPT_GRAPHIC_HEIGHT | graphicHeight |
| SCCOPT_GRAPHIC_HEIGHTLIMIT | graphicHeightLimit |
| SCCOPT_GRAPHIC_OUTPUTDPI | graphicOutputDPI |
| SCCOPT_GRAPHIC_SIZELIMIT | graphicSizeLimit |
| SCCOPT_GRAPHIC_SIZEMETHOD | graphicSizeMethod |
| SCCOPT_GRAPHIC_TRANSPARENCYCOLOR | graphicTransparencyColor |
| SCCOPT_GRAPHIC_WIDTH | graphicWidth |
| SCCOPT_GRAPHIC_WIDTHLIMIT | graphicWidthLimit |
| SCCOPT_IMAGEX_TIFFOPTIONS | tiffOptions |
| SCCOPT_IO_BUFFERSIZE | Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API:<br>readBufferSize<br>memoryMappedInputSize |

| Embedded API | SOAP API |
|---|---|
| | tempBufferSize |
| SCCOPT_JPEG_QUALITY | graphicJpegQuality |
| SCCOPT_MAXSSDBPAGEHEIGHT | maxSsDbPageHeight |
| SCCOPT_MAXSSDBPAGEWIDTH | maxSsDbPageWidth |
| SCCOPT_PRINTENDPAGE | endPage |
| SCCOPT_PRINTFONTALIAS | fontAlias |
| SCCOPT_PRINTSTARTPAGE | startPage |
| SCCOPT_RENDERING_PREFER_OIT | preferOITRendering |
| SCCOPT_REORDERMETHOD | reorderMethod |
| SCCOPT_SSPRINTDIRECTION | spreadsheetPageDirection |
| SCCOPT_SSPRINTFITTOPAGE | spreadsheetFitToPage |
| SCCOPT_SSPRINTGRIDLINES | spreadsheetShowGridLines |
| SCCOPT_SSPRINTHEADINGS | spreadsheetShowHeadings |
| SCCOPT_SSPRINTSCALEPERCENT | spreadsheetScalePercentage |
| SCCOPT_SSPRINTSCALEXHIGH | spreadsheetScaleXPagesHigh |
| SCCOPT_SSPRINTSCALEXWIDE | spreadsheetScaleXPagesWide |
| SCCOPT_TEMPDIR | NA |
| SCCOPT_TIMEZONE | timezone |
| SCCOPT_UNMAPPABLECHAR | unmappableCharacter |
| SCCOPT_USEDOCPAGESETTINGS | useDocumentPageSettings |
| SCCOPT_WHATTOPRINT | usePageRange |
| SCCOPT_WPEMAILHEADEROUTPUT | emailHeader |

## Search Export

This information pertains to Search Export.

| Embedded API | SOAP API |
|---|---|
| SCCOPT_DEFAULTINPUTCHARSET | defaultInputCharset |
| SCCOPT_ENABLEALLSUBOBJECTS | NA |
| SCCOPT_EXXML_DEF_METHOD | xmlDefinitionMethod |
| SCCOPT_EXXML_DEF_REFERENCE | xmlDefinitionLocation |
| SCCOPT_FALLBACKFORMAT | fallbackFormat |
| SCCOPT_FIFLAGS | extendedTestForText |
| SCCOPT_FILTERLZW | allowLZW |
| SCCOPT_FORMATFLAGS | isoDateTimes |
| SCCOPT_IO_BUFFERSIZE | Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API: |

| Embedded API | SOAP API |
|---|---|
| | readBufferSize<br>memoryMappedInputSize<br>tempBufferSize |
| SCCOPT_RENDERING_PREFER_OIT | preferOITRendering |
| SCCOPT_TEMPDIR | NA |
| SCCOPT_TIMEZONE | timezone |
| SCCOPT_UNMAPPABLECHAR | unmappableCharacter |
| SCCOPT_XML_DEF_METHOD | xmlDefinitionMethod |
| SCCOPT_XML_DEF_REFERENCE | xmlDefinitionLocation |
| SCCOPT_XML_NULLREPLACECHAR | nullReplacementCharacter |
| SCCOPT_XML_PAGEML_FLAGS | Each of the flags in the embedded option has a matching, stand-alone Boolean option in the SOAP API:<br>textOutOn<br>xmlDeclarationOff |
| SCCOPT_XML_PAGEML_PRINTERNAME | printerName |
| SCCOPT_XML_SEARCHML_CHAR_ATTR | Each of the flags in the embedded option has a matching, stand-alone Boolean option in the SOAP API:<br>allCapsOn<br>boldOn<br>doubleUnderlineOn<br>hiddenOn<br>italicOn<br>originalCharsetOn<br>outlineOn<br>revisionAddOn<br>revisionDeleteOn<br>smallCapsOn<br>strikeoutOn<br>underlineOn |
| SCCOPT_XML_SEARCHML_FLAGS | Each of the flags in the embedded option has a matching, stand-alone Boolean option in the SOAP API:<br>cellInfoOn<br>changeNumbertoTextOn<br>documentPropertiesOn<br>embeddingsOn<br>errorInfoOn<br>generateSystemData<br>metadataOnlyOn<br>paragraphStyleNamesOn<br>produceURLsOn<br>revisionsOn<br>suppressArchiveSubDocsOn |

| Embedded API | SOAP API |
|---|---|
| | suppressAttachmentsOn<br>xmlDeclarationOff |
| SCCOPT_XML_SEARCHML_OFFSET | includeTextOffsets |
| SCCOPT_XML_SEARCHML_PARA_ATTR | paragraphAttributes |
| SCCOPT_XML_SEARCHML_UNMAPPEDTEXT | unmappedText |

## HTML Export

This information pertains to HTML Export.

| Embedded API | SOAP API |
|---|---|
| SCCOPT_DEFAULTINPUTCHARSET | defaultInputCharset |
| SCCOPT_DEFAULTPRINTFONT | defaultFont |
| SCCOPT_EX_CALLBACKS | NA |
| SCCOPT_EX_CHANGETRACKING | showChangeTracking |
| SCCOPT_EX_CHARBYTEORDER | characterByteOrder |
| SCCOPT_EX_COLLAPSEWHITESPACE | collapseWhiteSpace |
| SCCOPT_EX_COMPLIANCEFLAGS | compliance |
| SCCOPT_EX_EXTRACTEMBEDDEDFILES | extractEmbeddedFiles |
| SCCOPT_EX_FALLBACKFONT | fallbackFont |
| SCCOPT_EX_FLAVOR | flavor |
| SCCOPT_EX_FONTFLAGS | fontFlags |
| SCCOPT_EX_GENBULLETSANDNUMS | genBulletsAndNums |
| SCCOPT_EX_GRIDADVANCE | gridAdvance |
| SCCOPT_EX_GRIDCOLS | gridCols |
| SCCOPT_EX_GRIDROWS | gridRows |
| SCCOPT_EX_GRIDWRAP | gridWrap |
| SCCOPT_EX_JAVASCRIPTTABS | javaScriptTabs |
| SCCOPT_EX_NOSOURCEFORMATTING | noSourceFormatting |
| SCCOPT_EX_OUTPUTCHARACTERSET | outputCharacterSet |
| SCCOPT_EX_PAGESIZE | pageSize |
| SCCOPT_EX_PREVENTGRAPHICOVERLAP | preventGraphicOverlap |
| SCCOPT_EX_SHOWHIDDENSSDATA | showHiddenSpreadsheetData |
| SCCOPT_EX_SHOWHIDDENTEXT | showHiddenText |
| SCCOPT_EX_SHOWSPREADSHEETBORDER | showSpreadsheetBorder |
| SCCOPT_EX_SIMPLESTYLENAMES | simpleStyleNames |
| SCCOPT_EX_SSDBBORDER | spreadsheetBorders |
| SCCOPT_EX_SSDBROWCOLHEADINGS | showSpreadsheetHeadings |

| Embedded API | SOAP API |
|---|---|
| SCCOPT_EX_TEMPLATE | template |
| SCCOPT_EX_UNICODECALLBACKSTR | NA |
| SCCOPT_FALLBACKFORMAT | fallbackFormat |
| SCCOPT_FIFLAGS | extendedTestForText |
| SCCOPT_FILTERJPG | allowJPEG |
| SCCOPT_FILTERLZW | allowLZW |
| SCCOPT_FORMATFLAGS | isoDateTimes |
| SCCOPT_GIF_INTERLACED | graphicGifInterlaced |
| SCCOPT_GRAPHIC_HEIGHTLIMIT | graphicHeightLimit |
| SCCOPT_GRAPHIC_OUTPUTDPI | graphicOutputDPI |
| SCCOPT_GRAPHIC_SIZELIMIT | graphicSizeLimit |
| SCCOPT_GRAPHIC_SIZEMETHOD | graphicSizeMethod |
| SCCOPT_GRAPHIC_TRANSPARENCYCOLOR | graphicTransparencyColor |
| SCCOPT_GRAPHIC_TYPE | graphicType |
| SCCOPT_GRAPHIC_WIDTHLIMIT | graphicWidthLimit |
| SCCOPT_IO_BUFFERSIZE | Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API:<br>readBuffermemory<br>MappedInputSize<br>tempBufferSize |
| SCCOPT_JPEG_QUALITY | graphicJpegQuality |
| SCCOPT_PRINTFONTALIAS | fontAlias |
| SCCOPT_RENDERING_PREFER_OIT | preferOITRendering |
| SCCOPT_REORDERMETHOD | reorderMethod |
| SCCOPT_TEMPDIR | NA |
| SCCOPT_TIMEZONE | timezone |
| SCCOPT_UNMAPPABLECHAR | unmappableCharacter |
| SCCOPT_WPEMAILHEADEROUTPUT | emailHeader |

# A

# SOAP Data Types and Options

This appendix defines the SOAP data types and options used by the export products.

This appendix contains the following sections:



## Simple Types

- xsd:base64Binary: Base64-encoded binary data.
- xsd:boolean: Binary data (true [non-zero] or false [0]).
- xsd:byte: Short data between -128 and 127.
- xsd:double: IEEE double-precision 64-bit floating point data.
- xsd:float: IEEE single-precision 32-bit floating point data.
- xsd:hexBinary: Arbitrary hex-encoded binary data.
- xsd:int: Long data between -2147483648 and 2147483647.
- xsd:short: Integer data between -32768 and 32767.
- xsd:signedInt: Integer data between -2147483648 and 2147483647.
- xsd:string: A null-terminated character string.
- xsd:unsignedByte: Unsigned, short data no greater than 255.
- xsd:unsignedInt: Unsigned, long data no greater than 4294967295.
- xsd:unsignedShort
- Unsigned, short data no greater than 65535.

## Complex Types

This section presents complex data types applicable to all products.

## IOSpec

This data type is a complexType structure that contains the full specification required for Transformation Server to open a particular data stream for input or output. In addition to a "specification", examples of which include a file system path or a URL, the structure also allows provides fields for the character set used in the specification and an identifier of the type of specification provided, for example, path or url. The IOSpec structure is defined as follows:

```
<complexType name="IOSpec">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="spec" type="ts:stringData" minOccurs="0" maxOccurs="1"/>
    <element name="specType" type="xsd:string" minOccurs="0" maxOccurs="1"
nillable="true"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

## stringData

This data type stores a text string along with an identifier of the character set used in the string. The charSet field indicates the character set used in the string. If the character set used in the string is UTF-8, the string may be passed to Transformation Server unmodified. If the string does not use the UTF-8 character set, the string must be passed in base64-encoded form. If the string is base64-encoded, the base64 field must be set to true.

This data type takes the form of a complexType structure, defined as follows:

```
<complexType name="stringData">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="str" type="xsd:string" minOccurs="0" maxOccurs="1"
nillable="true"/>
    <element name="charset" type="ts:CharacterSetEnum" minOccurs="0"
maxOccurs="1"/>
    <element name="base64" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

## stringList

StringList is an array of UTF-8 strings used with the fileAccess option. This data type allows passing lists of UNICODE strings (UTF-8 encoded) to and from Transformation Server. It does not support other UNICODE encodings, or non-UNICODE encodings.

This data type takes the form of a complexType structure, defined as follows:

```
<complexType name="StringList">
 <complexContent>
  <extension base="xsd:anySimpleType">
   <sequence>
    <element name="strings" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

### TransformResponse

This data type is a structure that contains a human-readable result message and a list of the output documents created by the transformation. The structure is defined as follows:

```
<complexType name="TransformResponse">
 <sequence>
  <element name="result" type="xsd:unsignedInt" minOccurs="0" maxOccurs="1"/>
  <element name="resultMsg" type="ts:stringData" minOccurs="0" maxOccurs="1"/>
  <element name="resultDocs" type="ts:ArrayOfIOSpec" minOccurs="0"
maxOccurs="1" nillable="true"/>
 </sequence>
</complexType>
```

## Enumerations

This section defines enumerations.

### CharacterByteOrderEnum

Valid for the characterByteOrder option.

#### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="CharacterByteOrderEnum">
 <restriction base="string">
  <enumeration value="big-endian"/>
  <enumeration value="little-endian"/>
  <enumeration value="template-order"/>
  </restriction>
</simpleType>
```

### CharacterSetEnum

Valid for the outputCharacterSet option.

#### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="CharacterSetEnum">
 <restriction base="string">
  <enumeration value="ISO-8859-1"/>
  <enumeration value="ISO-8859-2"/>
  <enumeration value="ISO-8859-3"/>
  <enumeration value="ISO-8859-4"/>
  <enumeration value="ISO-8859-5"/>
  <enumeration value="ISO-8859-6"/>
  <enumeration value="ISO-8859-7"/>
  <enumeration value="ISO-8859-8"/>
  <enumeration value="ISO-8859-9"/>
  <enumeration value="x-Mac-roman"/>
  <enumeration value="x-Mac-ce"/>
  <enumeration value="x-Mac-Greek"/>
  <enumeration value="x-Mac-Cyrillic"/>
  <enumeration value="x-Mac-Turkish"/>
  <enumeration value="GB2312"/>
```

```
  <enumeration value="Big5"/>
  <enumeration value="Shift_JIS"/>
  <enumeration value="KOI8-R"/>
  <enumeration value="windows-1250"/>
  <enumeration value="windows-1251"/>
  <enumeration value="windows-1252"/>
  <enumeration value="windows-1253"/>
  <enumeration value="windows-1254"/>
  <enumeration value="windows-1255"/>
  <enumeration value="windows-1256"/>
  <enumeration value="windows-1257"/>
  <enumeration value="EUC-KR"/>
  <enumeration value="EUC-JP"/>
  <enumeration value="ISO-2022-JP"/>
  <enumeration value="windows-874"/>
  <enumeration value="UTF-7"/>
  <enumeration value="UTF-8"/>
  <enumeration value="ISO-10646-UCS-2"/>
  <enumeration value="x-Charset-Unknown"/>
 </restriction>
</simpleType>
```

## ComplianceEnum

Valid for the compliance option.

### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="ComplianceEnum">
 <restriction base="string">
  <enumeration value="none"/>
  <enumeration value="well-formed"/>
  <enumeration value="strictDTD"/>
 </restriction>
</simpleType>
```

## DatabaseFitToPageEnum

Valid for the databaseFitToPage option.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="DatabaseFitToPageEnum">
 <restriction base="string">
  <enumeration value="dbNoScaling"/>
  <enumeration value="dbFitToPage"/>
  <enumeration value="dbFitToWidth"/>
  <enumeration value="dbFitToHeight"/>
 </restriction>
</simpleType>
```

## DefaultInputCharSetEnum

Valid for the defaultInputCharset option.

### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="DefaultInputCharSetEnum">
  <restriction base="string">
   <enumeration value="jis"/>
   <enumeration value="euc_jp"/>
   <enumeration value="cns11643_1"/>
   <enumeration value="euc_cns_1"/>
   <enumeration value="cns11643_2"/>
   <enumeration value="euc_cns_2"/>
   <enumeration value="ksc1987"/>
   <enumeration value="gb2312"/>
   <enumeration value="ebcdic37"/>
   <enumeration value="ebcdic273"/>
   <enumeration value="ebcdic274"/>
   <enumeration value="ebcdic277"/>
   <enumeration value="ebcdic278"/>
   <enumeration value="ebcdic280"/>
   <enumeration value="ebcdic282"/>
   <enumeration value="ebcdic284"/>
   <enumeration value="ebcdic285"/>
   <enumeration value="ebcdic297"/>
   <enumeration value="ebcdic500"/>
   <enumeration value="ebcdic1026"/>
   <enumeration value="ascii"/>
   <enumeration value="ansi437"/>
   <enumeration value="ansi737"/>
   <enumeration value="ansi850"/>
   <enumeration value="ansi852"/>
   <enumeration value="ansi855"/>
   <enumeration value="ansi857"/>
   <enumeration value="ansi860"/>
   <enumeration value="ansi861"/>
   <enumeration value="ansi863"/>
   <enumeration value="ansi865"/>
   <enumeration value="ansi866"/>
   <enumeration value="ansi869"/>
   <enumeration value="ansi874"/>
   <enumeration value="ansi932"/>
   <enumeration value="ansi936"/>
   <enumeration value="ansi949"/>
   <enumeration value="ansi950"/>
   <enumeration value="ansi1250"/>
   <enumeration value="ansi1251"/>
   <enumeration value="ansi1252"/>
   <enumeration value="ansi1253"/>
   <enumeration value="ansi1254"/>
   <enumeration value="ansi1255"/>
   <enumeration value="ansi1256"/>
   <enumeration value="ansi1257"/>
   <enumeration value="unicode"/>
   <enumeration value="iso8859_1"/>
   <enumeration value="iso8859_2"/>
   <enumeration value="iso8859_3"/>
   <enumeration value="iso8859_4"/>
   <enumeration value="iso8859_5"/>
   <enumeration value="iso8859_6"/>
   <enumeration value="iso8859_7"/>
   <enumeration value="iso8859_8"/>
   <enumeration value="iso8859_9"/>
   <enumeration value="macroman"/>
   <enumeration value="maccroatian"/>
   <enumeration value="macromanian"/>
   <enumeration value="macturkish"/>
   <enumeration value="macicelandic"/>
```

```
   <enumeration value="maccyrillic"/>
   <enumeration value="macgreek"/>
   <enumeration value="macce"/>
   <enumeration value="hebrew"/>
   <enumeration value="arabic"/>
   <enumeration value="macjis"/>
   <enumeration value="hproman8"/>
   <enumeration value="bidi_oldcode"/>
   <enumeration value="bidi_pc8"/>
   <enumeration value="bidi_e0"/>
   <enumeration value="htmlkoi8"/>
   <enumeration value="jis_roman"/>
   <enumeration value="utf7"/>
   <enumeration value="utf8"/>
   <enumeration value="littleendianunicode"/>
  </restriction>
</simpleType>
   <enumeration value="bigendianunicode"/>
```

## DefaultPageUnitsEnum

Valid for the defaultPageUnits option.

### Export Product

PDF Export

The enumeration is defined as follows:

```
<simpleType name="DefaultPageUnitsEnum">
 <restriction base="xsd:string">
  <enumeration value="inches"/>
  <enumeration value="points"/>
  <enumeration value="centimeters"/>
  <enumeration value="picas"/>
 </restriction>
</simpleType>
```

## DocumentMemoryModeEnum

Valid for the documentMemoryMode option.

### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

The enumeration is defined as follows:

```
  <simpleType name="DocumentMemoryModeEnum">
    <restriction base="string">
      <enumeration value="smallestMode"/>
      <enumeration value="smallMode"/>
      <enumeration value="mediumMode"/>
      <enumeration value="largeMode"/>
      <enumeration value="largestMode"/>
    <restriction>
  <simpleType>
```

## EmailHeaderOutputEnum

EmailHeaderOutputEnum takes the place of the MimeHeaderOutputEnum. Valid for the emailHeaderOutput option.

#### Export Product

HTML Export, Image Export, PDF Export

The enumeration is defined as follows:

```
<simpleType name="EmailHeaderOutputEnum">
  <restriction base="xsd:string">
    <enumeration value="emailHeaderStandard"/>
    <enumeration value="emailHeaderAll"/>
    <enumeration value="emailHeaderNone"/>
  </restriction>
</simpleType>
```

## ExtractEmbeddedFilesEnum

Valid for the extractEmbeddedFiles option.

#### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="ExtractEmbeddedFilesEnum">
 <restriction base="string">
  <enumeration value="ignoreFiles"/>
  <enumeration value="convertFiles"/>
  <enumeration value="extractFiles"/>
 </restriction>
</simpleType>
```

## FallbackFormatEnum

Valid for the fallbackFormat option.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="FallbackFormatEnum">
 <restriction base="string">
  <enumeration value="fallbackToText"/>
  <enumeration value="noFallbackFormat"/>
 </restriction>
</simpleType>
```

## FlavorEnum

Valid for the flavor option.

#### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="FlavorEnum">
 <restriction base="string">
  <enumeration value="generic-html"/>
  <enumeration value="generic-wireless-html"/>
  <enumeration value="html2.0"/>
  <enumeration value="html3.0"/>
  <enumeration value="html4.0"/>
```

```
  <enumeration value="netscape3.0"/>
  <enumeration value="netscape4.0"/>
  <enumeration value="internetExplorer3.0"/>
  <enumeration value="internetExplorer4.0"/>
  <enumeration value="avantGo3.3-palm"/>
  <enumeration value="avantGo3.3-palm-noTables"/>
  <enumeration value="avantGo3.3-winCE"/>
  <enumeration value="avantGo3.3-winCE-noTables"/>
  <enumeration value="webClipping1.1"/>
  <enumeration value="webClipping1.1-noTables"/>
  <enumeration value="chtml2.0"/>
  <enumeration value="hdml3.0"/>
  <enumeration value="text"/>
  <enumeration value="wml1.1"/>
  <enumeration value="wml1.1-withTables"/>
  <enumeration value="wml2.0"/>
  <enumeration value="xhtml-basic1.0"/>
  <enumeration value="xhtml-basic1.0-noTables"/>
 </restriction>
</simpleType>
```

## GraphicCroppingEnum

Valid for the graphicCropping option.

#### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="GraphicCroppingEnum">
   <restriction base="string">
   <enumeration value="noCropping"/>
   <enumeration value="cropToContent"/>
</restriction>
```

## GraphicSizeMethodEnum

Valid for the graphicSizeMethod option.

#### Export Product

HTML Export, Image Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="GraphicSizeMethodEnum">
 <restriction base="string">
  <enumeration value="smooth"/>
  <enumeration value="quick"/>
  <enumeration value="smoothGray"/>
 </restriction>
</simpleType>
```

## GraphicTypeEnum

Valid for the graphicType option.

#### Export Product

HTML Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="GraphicTypeEnum">
 <restriction base="string">
  <enumeration value="bmp"/>
  <enumeration value="gif"/>
  <enumeration value="jpeg"/>
  <enumeration value="noGraphics"/>
  <enumeration value="png"/>
  <enumeration value="wbmp"/>
 </restriction>
</simpleType>
```

## GraphicWatermarkScaleTypeEnum

Valid for the graphicWatermarkScaleType option.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="GraphicWatermarkScaleTypeEnum">
 <restriction base="string">
  <enumeration value="scaleWatermarkOff"/>
  <enumeration value="scaleWatermarkByPercent"/>
 </restriction>
</simpleType>
```

## GridAdvanceEnum

Valid for the gridAdvance option.

### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="GridAdvanceEnum">
 <restriction base="string">
  <enumeration value="advanceAcross"/>
  <enumeration value="advanceDown"/>
 </restriction>
</simpleType>
```

## MimeHeaderOutputEnum

Valid for the mimeHeaderOutput option. The mimeHeaderOutput option is no longer preferred, and has been replaced with the emailHeaderOutput option.

### Export Product

Image Export

The enumeration is defined as follows:

```
  <simpleType name="MimeHeaderOutputEnum">
    <restriction base="string">
      <enumeration value="all"/>
      <enumeration value="standard"/>
      <enumeration value="none"/>
    <restriction>
  <simpleType>
```

## oleEmbeddingsEnum

Valid for the oleEmbeddings option.

### Export Product

Search Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="OleEmbeddingsEnum">
 <restriction base="string">
  <enumeration value="processAll"/>
  <enumeration value="processNone"/>
  <enumeration value="processStandard"/>
 </restriction>
</simpleType>
```

## ReorderMethodEnum

Valid for the reorderMethod option.

### Export Product

HTML Export, Image Export, PDF Export, XML Export

The enumeration is defined as follows:

```
<simpleType name="ReorderMethodEnum">
 <restriction base="xsd:string">
  <enumeration value="reorderOff"/>
  <enumeration value="reorderLeftToRight"/>
  <enumeration value="reorderRightToLeft"/>
 </restriction>
</simpleType>
```

## SearchMLUnmappedTextEnum

Valid for the unmappedText option.

### Export Product

Search Export

The enumeration is defined as follows:

```
<simpleType name="SearchMLUnmappedTextEnum">
 <restriction base="string">
  <enumeration value="justUnmappedText"/>
  <enumeration value="noUnmappedText"/>
  <enumeration value="bothUnmappedText"/>
 </restriction>
</simpleType>
```

## SpreadSheetBordersEnum

Valid for the spreadsheetBorders option.

### Export Product

HTML Export

The enumeration is defined as follows:

```
<simpleType name="SpreadsheetBordersEnum">
 <restriction base="string">
  <enumeration value="createBorderIfMissing"/>
  <enumeration value="bordersOff"/>
  <enumeration value="useSourceBorders"/>
 </restriction>
</simpleType>
```

## SpreadsheetFitToPageEnum

Valid for the spreadsheetFitToPage option.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="SpreadsheetFitToPageEnum">
 <restriction base="string">
  <enumeration value="ssNoScaling"/>
  <enumeration value="ssFitToPage"/>
  <enumeration value="ssFitToWidth"/>
  <enumeration value="ssFitToHeight"/>
  <enumeration value="ssScaleByPercentage"/>
  <enumeration value="ssFitToPages"/>
 </restriction>
</simpleType>
```

## SpreadsheetPageDirectionEnum

Valid for the spreadsheetPageDirection option.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="SpreadsheetPageDirectionEnum">
 <restriction base="string">
  <enumeration value="downThenAcross"/>
  <enumeration value="acrossThenDown"/>
 </restriction>
</simpleType>
```

## TiffByteOrderEnum

Part of the TiffOptions structure.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="TiffByteOrderEnum">
 <restriction base="string">
  <enumeration value="little-endian"/>
  <enumeration value="big-endian"/>
 </restriction>
</simpleType>
```

## TiffColorSpaceEnum

Part of the TiffOptions structure.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="TiffColorSpaceEnum">
 <restriction base="string">
  <enumeration value="blackWhite-1Bit"/>
  <enumeration value="palette-8Bit"/>
  <enumeration value="rgb-24Bit"/>
 </restriction>
</simpleType>
```

## TiffCompressionEnum

Part of the TiffOptions structure.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="TiffCompressionEnum">
 <restriction base="string">
  <enumeration value="noCompression"/>
  <enumeration value="packbits"/>
  <enumeration value="LZW"/>
  <enumeration value="CCITT-1D"/>
  <enumeration value="CCITT-Group3-Fax"/>
  <enumeration value="CCITT-Group4-Fax"/>
 </restriction>
</simpleType>
```

## TiffFillOrderEnum

Part of the TiffOptions structure.

### Export Product

Image Export

The enumeration is defined as follows:

```
<simpleType name="TiffFillOrderEnum">
 <restriction base="string">
  <enumeration value="fillOrder-1"/>
  <enumeration value="fillOrder-2"/>
 </restriction>
</simpleType>
```

## WatermarkPositionEnum

Valid for the watermarkPosition option.

### Export Product

PDF Export

The enumeration is defined as follows:

```
<simpleType name="WatermarkPostionEnum">
 <restriction base="xsd:string">
  <enumeration value="centerOfPage"/>
 </restriction>
</simpleType>
```

### WatermarkScalingEnum

Valid for the watermarkScaling option.

#### Export Product

PDF Export

The enumeration is defined as follows:

```
<simpleType name="WatermarkScalingEnum">
 <restriction base="xsd:string">
  <enumeration value="pdfNoMap"/>
  <enumeration value="pdfFitToPage"/>
  <enumeration value="pdfScale"/>
 </restriction>
</simpleType>
```

### XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option.

#### Export Product

Search Export

The enumeration is defined as follows:

```
<simpleType name="XmlDefinitionMethodEnum">
 <restriction base="string">
  <enumeration value="dtd"/>
  <enumeration value="noDefinition"/>
  <enumeration value="xsd"/>
 </restriction>
</simpleType>
```

## SOAP Options

This section details the Web Services implementation of options in Transformation Server. However, there are references to API-specific information for the C and JAVA client interfaces to the technology within each of the following sections.

### How Options Work

An option is defined by an identifier and an associated value. The identifier (hOptions) indicates what particular option is being specified. The option value data must be in a form that conforms to the set of supported data types.

Note that it is not necessarily an error to specify options that are not understood by the export engine, but some transformation engines may require that certain options be specified.

### Character Mapping

This section applies to character mapping options.

### 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.

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

This option supersedes the 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 fallbackFormat, though internally such values will be translated into equivalent values for the defaultInputCharset. As a result, if an application were to set both options, the last such value set for either option will be the value that takes effect.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

DefaultInputCharSet

#### Data

The SOAP representation of the character set to use, from the values in defaultInputCharSetEnum.

### characterByteOrder

This option determines the byte order of Unicode characters in the output files when Unicode is chosen as the output character set.

#### Export Product

HTML Export

#### Data Type

CharacterByteOrderEnum

#### Data

One of the following values:

- big-endian: Big-Endian byte ordering is common on RISC and Motorola processors. The ISO 10646 standard, the Unicode Standard and the W3C recommend Big-Endian Unicode. It also corresponds to network byte order.
- little-endian: Little Endian is common on Intel processors.
- template-order: This value will cause the output to use the byte ordering used in the main template file, if the template is written in Unicode. If the template is not written in Unicode, Big-Endian byte order is used.

#### Default

template-order

#### Equivalents

- C Client Implementation: OIT_CharacterByteOrderEnum
- JAVA Client Implementation: CharacterByteOrderEnum

### outputCharacterSet

This option allows the developer to specify which character set should be used in the output file. The technology will then translate or "map" characters from the input document's character set to the output character set as needed. Naturally, export process does not translate content from one language to another. This character mapping is also clearly limited by the need for the character to be in both the input and the output character sets. If a character cannot be mapped, the character will show up in the output as the "unmappable character." The default unmappable character used is the "*" character. The character used may be changed by setting the unmappableCharacter option. If the resulting output contains an excessive number of these "*" characters, selecting a more appropriate output character set should improve the situation.

The technology reserves the right to override this option. The option will be overridden if ANSI Double-Byte Character Set (DBCS) characters are detected in the source document and a single-byte character set is chosen as the output character set. If the option is overridden, this change will affect the entire output document. The technology uses the first DBCS character set it finds in the document as the basis for its decision about which output character set to choose as its override.

Note that special character set override rules apply when the input document uses the HWP (Hangul 97) filter. For these documents, the output character set will be forced to EUC-KR unless the user has selected euc-kr, Unicode or UTF-8 output. These override rules do not apply to the HWP2 (Hangul 2002) filter, as it uses Unicode exclusively.

Source documents in Unicode will not override this option. This is especially important to remember as some important file formats store text in Unicode including Microsoft Office.

The markup standards currently supported by HTML Export limit documents to a single character set. That character set is specified in an output file using the CONTENT attribute of the <meta> tag. This limits what the technology can do with documents that have multiple character sets. In general, documents that are a mix of a single Asian language and English characters will translate correctly (although with some possible loss of non-alphanumeric characters) if the appropriate DBCS, UTF-8 or Unicode output character set is selected. This is because most DBCS character sets include the standard 7-bit Latin 1 characters. Documents that contain more than one DBCS character set or a DBCS character set and a non-English character set (such as Cyrillic) may not export with all the character glyphs intact unless Unicode or UTF-8 is used.

While the W3C recommends using Unicode, there is a downside to it at this time. Not all systems have the appropriate fonts needed for using Unicode or UTF-8. Many editors do not understand these character sets, as well. In fact, while HTML Export can read Unicode source documents, it cannot read UTF-8 source documents. In addition, there are some differences in the way browsers interpret the byte order of 2-byte Unicode characters. For additional details about the byte ordering issue, see characterByteOrder.

An additional HTML browser idiosyncrasy affects the Netscape 4.0 – 6.0 browsers. While these browsers properly render Unicode HTML, they seem to be unable to read .css files that are written in Unicode. For this reason, if the output character set is Unicode and the HTML flavor (described in flavor) being generated is Netscape 4.0 or the common 4.0 flavor, the associated .css file will be written in UTF-8.

In order for HTML Export to correctly place the character set into the output file it generates, all templates should include a statement that uses the `{## insert}` macro to insert the character set into the document, as in the following example:

```
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset={## insert element=pragma.charset}" />
```

If the template does not include this line, the user may have to manually select the correct character set in the user's browser.

### Export Product

HTML Export

### Data Type

CharacterSetEnum

### Data

One of the following values:

| Value | Text used in <META…> tag | Description |
|---|---|---|
| ISO-8859-1 | iso-8859-1 | Latin-1 - this is a subset of Windows 1252 |
| ISO-8859-2 | iso-8859-2 | Latin-2 |
| ISO-8859-3 | iso-8859-3 | Latin-3 |
| ISO-8859-4 | iso-8859-4 | Latin-4 |
| ISO-8859-5 | iso-8859-5 | Cyrillic |
| ISO-8859-6 | iso-8859-6 | Arabic |
| ISO-8859-7 | iso-8859-7 | Greek |
| ISO-8859-8 | iso-8859-8 | Hebrew |
| ISO-8859-9 | iso-8859-9 | Turkish |
| x-Mac-roman | x-mac-roman | Mac Roman |
| x-Mac-ce | x-mac-ce | Mac CE |
| x-Mac-Greek | x-mac-greek | Mac Greek |
| x-Mac-Cyrillic | x-mac-cyrillic | Mac Cyrillic |
| x-Mac-Turkish | x-mac-turkish | Mac Turkish |
| GB2312 | gb2312 | Simplified Chinese |
| Big5 | big5 | Traditional Chinese |
| Shift_JIS | Shift_JIS | Japanese |
| KOI8-R | koi8-r | Russian |
| windows-1250 | x-cp1250 | Eastern Europe |

| Value | Text used in <META…> tag | Description |
|---|---|---|
| windows-1251 | x-cp1251 | Cyrillic |
| windows-1252 | windows-1252 | Western - Windows 1252 |
| windows-1253 | windows-1253 | Greek |
| windows-1254 | windows-1254 | Turkish |
| windows-1255 | windows-1255 | Hebrew |
| windows-1256 | windows-1256 | Arabic |
| windows-1257 | windows-1257 | Baltic |
| EUC-KR | euc-kr | Korean Hangul KSC 5601-1987 Wansung |
| EUC-JP | euc-jp | Japanese EUC |
| ISO-2022-JP | iso-2022-jp | JIS (Japanese) |
| windows-874 | windows-874 | Thai |
| UTF-8 | UTF-8 | UTF-8 (a Unicode variant) |
| ISO-10646-UCS-2 | ISO-10646 | Unicode |

#### Default

- windows-1252

#### Equivalents

- C Client Implementation: TS_CharacterSetEnum
- JAVA Client Implementation: CharacterSetEnum

### unmappableCharacter

This option selects the character used when a character cannot be found in the output character set. This option takes the Unicode value for the replacement character. It is left to the user to make sure that the selected replacement character is available in the output character set.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:unsignedShort

#### Data

The Unicode value for the character to use.

#### Default

- 0x002a = "*"

#### Equivalents

- C Client Implementation: XSD_unsignedShort
- JAVA Client Implementation: UnsignedShort

## Output

This section applies to output options.

### altlink

The option takes the form of a data structure that contains two ts:stringData structures; one for the "prev" link and one for the "next" link. These values are used in the transformation process when it is creating multiple output files that link to and from each other. The "prev" altlink is used for the link-to-previous-item in the first output file. The "next" altlink is used for the link-to-next link on the last page of output.

#### Export Product

HTML Export

#### Data Type

The AltLink data type is defined as follows:

```
<complexType name="AltLink">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="prev" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <element name="next" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

#### Data

The altlink option is a complexType data structure composed of ts:stringData values. The values are links to the "prev" and "next" output files.

#### Default

These strings are empty by default.

#### Equivalents

- C Client Implementation: OIT_AltLink
- JAVA Client Implementation: AltLink

### showChangeTracking

The setting for this option determines whether or not change tracking information in input documents will be written into the output via the <ins> and <del> HTML tags. When the option is set to false, no change tracking information will be written into the output. When set to true, the <ins> and <del> tags will be used as appropriate.

Previous versions of HTML Export included change tracking text in comments.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### collapseWhiteSpace

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

When set, this option deletes whitespace from the output document. Two types of whitespace are removed: redundant whitespace characters and vertical whitespace. This option is intended for situations where bandwidth and screen space are limited.

The HTML standard specifies that the browser will collapse a sequence of whitespace characters into a single whitespace character. Therefore, having HTML Export remove these redundant whitespace characters has no effect on the final view of the document. Removing them benefits the document in reducing the overall size of the output files generated and thereby saves bandwidth and decreases file transmission times. While HTML Export makes an effort to remove as much redundant whitespace as possible, there will be cases where some extra spacing appears in the output.

Removing vertical whitespace, on the other hand, does affect the look of the document in the browser. When possible, HTML Export preserves vertical spacing between elements. However, when this option is set, vertical whitespace is removed, resulting in a more compact view.

Please note that the collapse white space option does not affect whitespace coming from the template.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true Whitespace is removed.
- false: Whitespace is left intact.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### compliance

This option allows the developer to force the output to be compliant with a given standard. Currently, only DTD and well-formed compliance are supported. The option takes the form of a set of bit flags for toggling the available options. Flags are off by default and are turned on by bitwise OR-ing them together.

#### Export Product

HTML Export

#### Data Type

ComplianceEnum

#### Data

Any of the following flags:

- strictDTD: Set to enforce strict DTD compliance in the HTML written. The resulting HTML will be well formed. This means that an XML parser can parse it. In addition, "safe" HTML tags normally written by HTML Export are turned off when this flag is set. For more information about "safe" tags, see flavor.

  Especially when using older HTML flavors, use of this flag somewhat diminishes the fidelity of the view of the output document compared to the original document. In addition to other changes to the output, setting this flag also has the same effect as setting the preventGraphicOverlap option to true.

  This flag should not be used with the well-formed flag. If they are both set, this flag will override the well-formed flag.

  Most users will probably want to use the well-formed flag instead of this flag.

- well-formed: Set to force the HTML written to be well formed. This means that an XML parser can parse it. This option differs from the strictDTD flag in that it allows "safe" tags. This flag should not be used with the strictDTD flag. If they are both set, the strict DTD flag will override this flag. For most users, this flag is recommended over the use of the strictDTD flag as it produces well formed, XHTML compliant HTML without the penalties imposed by the strict DTD flag.

- none: All flags turned off.

#### Default

- none

#### Equivalents

- C Client Implementation: OIT_ComplianceEnum
- JAVA Client Implementation: ComplianceEnum

### extractEmbeddedFiles

This option controls the extraction of embedded documents in the input document. When set to extractFiles, the embedding will be extracted in its native format, allowing it to be read by the authoring application. When set to convertFiles, the embedding will be extracted as HTML. When set to ignoreFiles, the embedding will be ignored.

This option is only valid for UUE, MIME and MSG files and not for general purpose file attachments.

#### Export Product

HTML Export

#### Data

- ignoreFiles: Embeddings are skipped.
- convertFiles: Embeddings are converted.
- extractFiles: Embeddings are extracted in their native file format.

#### Default

ignoreFiles

#### Equivalents

- C Client Implementation: OIT_ExtractEmbeddedFilesEnum
- JAVA Client Implementation: ExtractEmbeddedFilesEnum

### flavor

Each Web browser forms a de facto HTML standard. This is because each browser has a unique collection of HTML tags and tag attributes it does or does not support. Thus, there are a large number of browser-based variations on the official HTML standards that are referred to here as "flavors" of HTML.

This option allows the developer to tailor the output generated to a specific browser or for a specific minimum browser. This allows HTML Export to produce the best possible rendering of the source document given the tags available in the target flavor. It also gives the OEM the ability to specify which standard their product will adhere to, rather than having that standard be dictated by HTML Export.

HTML Export currently supports a large number of flavors. While some flavors are targeted at specific browsers, other flavors are designed for a more abstract target. The "generic" and "HTML 2.0" flavors provide "lowest common denominator" flavors. The HTML produced by these flavors is very simple and should work in almost any browser. The primary difference between these two flavors is that the generic flavor supports tables and the HTML 2.0 flavor does not.

At other times, it is desirable to have the ability to create HTML that simply supports "the major x.0 and later browsers." For this purpose, there are the "greatest common

denominator" flavors. They are the "3.0" and "4.0" flavors. The "3.0" flavor should be used to create HTML that will look good in Netscape Navigator 3.0 or later and in Microsoft Internet Explorer 3.0 or later. The "4.0" flavor is defined to look good in Netscape Navigator 4.0 or later and in Microsoft Internet Explorer 4.0 or later. Note that upon examining the capabilities of these browsers after the 4.0 versions, it was determined that while they offer many new features, they do not have any .html or .css extensions that are useful to HTML Export at this time.

Naturally, support for a particular HTML flavor does not mean that HTML Export will generate all the tags and tag attributes that flavor supports. There are many tags and attributes that cannot sensibly be used in an automated conversion setting. Such tags require more information about the author's intent than is available in the source document.

Exporting a document to a particular HTML flavor also does not mean that the resulting HTML will be limited to only the tags and tag attributes supported by that flavor. In many cases, HTML Export will write out extra "safe" tags to the document, unless compliance (see compliance) has the strictDTD flag set. The target browser will safely ignore this extra HTML. However, should the converted document be viewed in a more sophisticated browser, this extra information will be used to produce a more accurate view of the document.

What support for a particular HTML flavor does mean is that the HTML generated will look as good as possible when viewed in the appropriate browser.

### Export Product

HTML Export

### Data Type

FlavorEnum

### Data

One of the following values. Note that the flavors marked with "(CSS)" indicate that the flavor in question requires the creation of a separate or embedded .css file as part of the document conversion.

| Value | Description |
|---|---|
| generic-html | General purpose, simple HTML support that should look good in any browser that supports tables. |
| html2.0 | HTML 2.0. Based on the official HTML 2.0 standard, this provides minimal HTML support and per that standard, it does not support tables. |
| html3.0 | Should look good in both Netscape Navigator 3.0 or later and Microsoft Internet Explorer 3.0 or later. |
| html4.0 | Should look good in both Netscape Navigator 4.0 or later and Microsoft Internet Explorer 4.0 or later (CSS). |
| netscape3.0 | Netscape Navigator 3.0 |
| netscape4.0 | Netscape Navigator 4.0 (CSS) |
| internetExplorer3.0 | Microsoft Internet Explorer 3.0. Note that while this flavor has limited CSS support, it does not create a separate or embedded .css file. |
| internetExplorer4.0 | Microsoft Internet Explorer 4.0 (CSS) |

#### Default

- html4.0

#### Equivalents

- C Client Implementation: OIT_FlavorEnum
- JAVA Client Implementation: FlavorEnum

### noSourceFormatting

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

This option turns off writing of characters that are produced strictly to make the output more readable and visually appealing. Currently, those formatting characters are limited to newlines, carriage returns and spaces. This option is of benefit primarily to users who perform special automated processing on the text produced by the technology. For these users, even benign non-markup text not originally in the source document constitutes a source of extra headaches for their processing. Setting this option excludes all formatting characters from appearing in the generated markup.

It is important to note the things that setting this option does not do:

- While setting this option will make it very difficult for a human to read the generated markup in a text editor, it does not affect the browser's rendering of the document.
- This option does not affect the contents of the .css files since they do not contain any text from the source document.
- The option does not affect spaces or newlines copied from the template as the contents of the templates are already under the control of the customer.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true: Do not output formatting characters.
- false: Include formatting characters in the output.

#### Default

- false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### showHiddenSpreadsheetData

The setting for this option determines whether or not hidden data (hidden columns, rows or sheets) in a spreadsheet will be included in the output. When set to false (the default), the hidden elements are not written. When set to true, they are placed in the output in the same manner as regular spreadsheet data.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

- true: Allow hidden data to be placed in the output.
- false: Prevent hidden data from being placed in the output.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### showHiddenText

This option will force HTML Export to place all hidden text in line with surrounding text.

Please note that enabling this option will not display hidden cells, hidden rows or hidden sheets in spreadsheet documents. Also note that when graphic documents (such as faxes) are processed by OCR software and converted to PDF, the optically recognized text may be rendered as a layer of hidden text behind the original image. In order to properly export such PDF documents, this option must be enabled.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

- true: Allow hidden text to be placed in the output.
- false: Prevent hidden text from being placed in the output.

### Default

false

### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## simpleStyleNames

This option is for use by people who intend to read or change the CSS style names generated by HTML Export.

By default, HTML Export creates unique style names based on the style names used in the original document. Unfortunately, there is an inherent limitation in the style names the CSS standard permits. That standard only permits the characters [a-z][A-Z][0-9] and "-". Source document style names do not necessarily have this restriction. In fact they may even contain Unicode characters at times. For this reason, the original style names may need to be modified to conform to this standard. To avoid illegal style names, HTML Export performs the following substitutions on all source style names:

1. If the character is a "-", then it is replaced with "--".
2. If the character is not one of the remaining characters ([a-z][A-Z][0-9]), then it is replaced by "-xxxx" where "xxxx" is the Unicode value of the character in hexadecimal.
3. Otherwise the character appears in the style name normally.

An example of one of the most common examples of this substitution is that spaces in style names are replaced with "-0020". For a more complete example of this character substitution in style names, consider the source style name *My Special H1-Style!*. This would be transformed to:

```
My-0020Special-0020H1--Style-0021
```

While admittedly this system lacks a certain aesthetic, it avoids the problem of how the document looks when the browser receives duplicate or invalid style names. Developers should also appreciate the simplicity of the code needed to parse or create these style names.

In addition, HTML Export will sometimes create special character attribute-only versions of styles. These have the same name as the style they are based on with "--Char" appended to the end. These styles differ from their original counterparts in that they contain no block level CSS. This more general solution replaces the solution implemented in versions 7.1 and earlier which created "--List" styles to solve a subset of this problem. This was done to work around limitations in some browsers.

Because of these CSS limitations, the simpleStyleNames (see simpleStyleNames) option was created. Setting this option to true causes HTML Export to generate style names that are easy to read but are not guaranteed to be unique. It does this by discarding all characters in the original style name that are not legal in CSS style names. As one would expect, this may lead to naming collisions.

An example of a naming collision caused by setting this option can be seen if you look at source document styles named MyStyle and My $ Style. When exported with this option, both would become MyStyle. This in turn may generate confusion when

viewing the document in the browser. This is because the browser will look upon the second style as being a redefinition of the first.

With the option set to false this is not a problem. The two styles would be converted to MyStyle and My-0020-0024-0020Style respectively. Because the style names are unique, the browser will not see the second style as a redefinition of the first.

As this contrived example indicates, naming collisions should be rare for most U.S. documents.

If a style name consists of nothing but illegal characters, HTML Export will create a style name for it. This style name is of the form UnnamedStyleX where "X" is a count of styles encountered so far that did not have style names for one reason or another. This behavior is expected to be very common when converting international documents in languages that are not based on 7-bit ASCII.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true: Generate names that may not be unique, but are easy to read.
- false: Generate unique style names that are difficult to read.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### preferOITRendering

This option is only valid on the Linux (Red Hat and Suse) and Solaris Sparc 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). This requires that there be an X11 display and a valid DISPLAY variable, regardless of the type of input document.

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, Image Export will exit with an error. Also note that when copying Windows fonts to a UNIX system, the font extension for the files (*.ttf or *.ttc) must be lowercase, or they will not

be detected during the search for available fonts. Oracle does not provide fonts with any Outside In product.

If preferOITRendering is set in a particular instance of tsagent, it cannot be changed in that agent until the agent is terminated.

#### Export Product

Image Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### 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

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## Input Handling

This section discusses input handling options.

### 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 noFallbackFormat be set to prevent Image Export from exporting unidentified binary files as though they were text, which could generate many pages of "garbage" output.

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 defaultInputCharset option for such functionality.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

FallbackFormatEnum

#### Data

One of the following values:

- fallbackToText: Unidentified file types will be treated as text files.
- noFallbackFormat: 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 SCCERR_UNSUPPORTEDFORMAT.

#### Default

- ASCII-8

#### Equivalents

- C Client Implementation: OIT_FallbackFormatEnum
- JAVA Client Implementation: FallbackFormatEnum

### extendedTestForText

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.)

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

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

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### ignorePassword

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.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### Data

- true: Ignore validation of the password
- false: Prompt for the password

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### oleEmbeddings

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 processAll option is required. Post Office-2003 products such as Office 2007 embeddings also fall into this category.

---

#### Export Product

Search Export, XML Export

#### Data Type

OleEmbeddingsEnum

#### Data

- processAll: Process all embeddings in the file.
- processNone: Process none of the embeddings in the file
- processStandard: Process embeddings that are known standard embeddings.

#### Default

processStandard

#### Equivalents

- C Client Implementation: OIT_OleEmbeddingsEnum
- JAVA Client Implementation: OleEmbeddingsEnum

### 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).

#### Export Product

HTML Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### Data

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

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### reorderBIDI

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.

#### Export Product

HTML Export, Image Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### Data

- true: The PDF filter uses standard ordering.
- false: The PDF filter will attempt to reorder bidirectional text runs.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### skipLinkedImages

This option allows the developer to choose how links to images in input files should be handled.The developer may request that the link be handled in one of two different ways:

- Have HTML Export attempt to follow the link, convert it to the selected image type, and insert the converted object into the output (this is the default behavior).
- Ignore the link altogether.

When set to true, this option will skip linked images when processing output files. If set to false, linked images will be converted and included in the output.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

- true: Skip linked images
- false: Include linked images in the output

#### Default

false

#### Equivalents

C Client Implementation: XSD_boolean

JAVA Client Implementation: Boolean

### 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 (e.g., most dates in spreadsheet cells). This option will not affect dates that are stored as text.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:int

#### 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 the numeric value of 61440 (0xF000 in hexadecimal).

#### Default

- 0: GMT time

#### Equivalents

- C Client Implementation: XSD_int
- JAVA Client Implementation: Integer

### 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.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Data

- true
- false

**Default**

- false

### htmlCondCommentIE5On

This option allows you to display content customized for Internet Explorer 5.

**Export Product**

HTML Export, Image Export, PDF Export, Search Export, XML Export

**Data Type**

xsd_boolean

**Default**

0: off

**Equivalents**

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### htmlCondCommentIE6On

This option allows you to display content customized for Internet Explorer 6.

**Export Product**

HTML Export, Image Export, PDF Export, Search Export, XML Export

**Data Type**

xsd_boolean

**Default**

0: off

**Equivalents**

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### htmlCondCommentIE7On

This option allows you to display content customized for Internet Explorer 7.

**Export Product**

HTML Export, Image Export, PDF Export, Search Export, XML Export

**Data Type**

xsd_boolean

#### Default

0: off

#### Equivalents

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### htmlCondCommentIE8On

This option allows you to display content customized for Internet Explorer 8.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd_boolean

#### Default

0: off

#### Equivalents

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### htmlCondCommentIE9On

This option allows you to display content customized for Internet Explorer 9.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd_boolean

#### Default

0: off

#### Equivalents

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### htmlCondCommentAllOn

This option allows you to display all conditional comments.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd_boolean

#### Default

0: off

#### Equivalents

C Client Implementation: VTBOOL

JAVA Client Implementation: boolean

### pdfFilterDropHyphens

This option controls whether or not the PDF filter will drop hyphens at the end of a line. Since most PDF-generating tools create them as generic dashes, it's impossible for Outside In to know if the hyphen is a syllable hyphen or part of a hyphenated word. When this option is set to TRUE, all hyphens at the end of lines will be dropped from the extracted text.

---

**Note:**

When this option is TRUE, the character counts for the extracted text may not match the counts used for rendering where the hyphens are required for rendering. This will affect annotations in rendering APIs.

---

#### Export Product

HTML Export

#### Data Type

xsd_boolean

#### Data

- TRUE: This setting drops hyphens from the end of all lines.
- FALSE: This setting retains hyphens at the end of all lines.

#### Default

FALSE

#### Equivalents

- C Client Implementation: VTBOOL
- JAVA Client Implementation: boolean

## Layout

This section discusses layout options.

### fallbackFont

Determines what font will be used when the font specified by the document is not available.

Currently this option is only used in certain situations where a CSS flavor of HTML is in use. Specifically, this option helps to avoid problems in some browsers where symbol fonts like Wingdings are used for the bullets in lists, and the body of the list is in a font the browser cannot find. In this case, specifying a fallback font prevents the browser from using/cascading the Wingdings font into the text of the list when the browser cannot find the font specified for the list text.

To turn off the fallback font, this option must be explicitly set to an empty string (""). While turning off the fallback font is not recommended, it will result in a minor reduction in the size of the HTML and CSS generated.

#### Export Product

HTML Export

#### Data Type

stringData

#### Data

The name of the fallback font and the character set of that font.

#### Default

If this option is not set, "Arial" is used as the default fallback font.

#### Equivalents

- C Client Implementation: TS_stringData
- JAVA Client Implementation: StringData

### fontFlags

This option is used to turn off specified font-related markup in the output. Naturally, if the requested output flavor or other option settings prevent markup of the specified type from being written, this option cannot be used to turn it back on. However, specifying the size, color and font face of characters may all be suppressed by bitwise OR-ing together the appropriate combination of flags in this option.

#### Export Product

HTML Export

#### Data Type

The FontFlags option takes the form of a data structure, defined as follows:

```
<complexType name="FontFlags">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="suppressSize" type="xsd:boolean" minOccurs="0"
```

```
maxOccurs="1"/>
    <element name="suppressColor" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
    <element name="suppressFace" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

### Data

The FontFlags option is a complexType data structure composed of the following Boolean variables, which may be switched on or off in any combination:

- suppressSize: When switched on, turns off any character-sizing information supported in the output flavor. As an example, this flag could be useful when exporting presentation files where the author specified a very large font.
- suppressColor: When switched on, suppresses specifying the color of text. This is particularly useful for exports of documents containing white text.
- suppressFace: When switched on, prevents the technology from requesting a specific font (e.g. "Arial", "Courier", etc.) name for text. This can be useful if the template author feels that the original document font is not likely to be available to those who are viewing the converted document.

### Default

All flags set to false, in which case no font information is suppressed.

### Equivalents

- C Client Implementation: OIT_FontFlags
- JAVA Client Implementation: FontFlags

## genBulletsAndNums

Turning this option on causes the technology to generate list numbers and/or bullets as needed rather than using list markup tags. While this violates the spirit of what markup languages should do, it does cause the browsers to render the lists in a way that is more faithful to the original look of the document. Figure 1 is an example of a list that does not view well with this option turned off.

1. Item 1

1.1 Item 1.1

1.1.1 Item 1.1.1

1 Item 1

1 Item 1.1

1 Item 1.1.1

This is due to the way browsers render <li> tags. The HTML standards currently do not allow any way to specify outline style list numbering.

One limitation when using this option is that standard list indentation may not be possible due to the limits of the selected output HTML flavor. At this time, only the

HTML flavors where CSS is available support the kind of hanging indents normally associated with lists.

If a bullet character needs to be generated, Unicode character 0x2022 will be used. Note that many character sets do not contain this character, so the unmappable character ("*") would be used in that case.

***Figure 1 List Example***

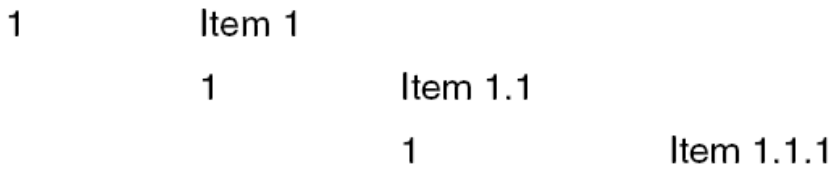
1. Item 1

1.1 Item 1.1

1.1.1 Item 1.1.1

Figure 2 is an example of how today's most popular browsers would render the preceding list.

***Figure 2 Browser-rendered List***




This is due to the way browsers render <li> tags. The HTML standards currently do not allow any way to specify outline style list numbering.

One limitation when using this option is that standard list indentation may not be possible due to the limits of the selected output HTML flavor. At this time, only the HTML flavors where CSS is available support the kind of hanging indents normally associated with lists.

If a bullet character needs to be generated, Unicode character 0x2022 will be used. Note that many character sets do not contain this character, so the unmappable character (“*") would be used in that case.

### Export Product

HTML Export

### Data Type

xsd:boolean

### Data

One of the following values:

- true: Generate list item numbers and bullets.
- false: Use list markup tags.

### Default

- false

### Equivalents

C Client Implementation: XSD_boolean

JAVA Client Implementation: Boolean

### gridAdvance

Options related to grids have no effect on the output unless a template that has been enabled with the `{## unit}` template macro is in use.

This option allows the developer to specify how the "previous" and "next" relationships will work between grids. As such, it is only useful when a grid-enabled template has been selected with the template (see template) option.

Setting this option to advanceAcross causes the grids.next.body template element to traverse the input spreadsheet or database by rows:

Grid 1 Grid 2 Grid 3

Grid 4 Grid 5 Grid 6

Grid 7 Grid 8 Grid 9

Setting this option to advanceDown causes the grids.next.body template element to traverse the input spreadsheet or database by columns:

Grid 1 Grid 4 Grid 7

Grid 2 Grid 5 Grid 8

Grid 3 Grid 6 Grid 9

Figure 3 shows how setting this option to advanceAcross causes the grids.next.body template element to traverse the input spreadsheet or database by rows.

***Figure 3 Traverse Input By Rows***

| | | |
|---|---|---|
| Grid 1 | Grid 2 | Grid 3 |
| Grid 4 | Grid 5 | Grid 6 |
| Grid 7 | Grid 8 | Grid 9 |

Figure 4 shows how setting this option to advanceDown causes the grids.next.body template element to traverse the input spreadsheet or database by columns.

***Figure 4 Traverse Input By Columns***

| | | |
|---|---|---|
| Grid 1 | Grid 4 | Grid 7 |
| Grid 2 | Grid 5 | Grid 8 |
| Grid 3 | Grid 6 | Grid 9 |

### Export Product

HTML Export

### Data Type

GridAdvanceEnu

### Data

One of the following values:

- advanceAcross: To traverse by rows.
- advanceDown: To traverse by columns.

### Default

- advanceDown

### Equivalents

- C Client Implementation: OIT_GridAdvanceEnum
- JAVA Client Implementation: GridAdvanceEnum

## gridCols

Options related to grids have no effect on the output unless a template that has been enabled with the `{## unit}` template macro is in use.

This option allows the developer to specify the number of columns that each template "grid" (applicable only to spreadsheet or database files) should contain. As such, it is only useful when a grid-enabled template has been selected with the template (see template) option.

Setting this option to zero ("0") means that no limit is placed on the number of columns in the grid. However, the settings of the pageSize, gridCols and gridRows options must all be taken into account when determining the actual dimensions of the grids used during an export. The following table describes the interaction of these options when a template is using grids:

| Grid Row/Col Value | Page Size is 0 | Page Size is not 0 |
|---|---|---|
| Grid Rows and Grid Cols are both 0. | The entire spreadsheet is exported. | The grid size is determined based on the Page Size. |
| Grid Rows is 0. Grid Cols is not 0 or the default value. | The table is broken into grids that are Grid Cols wide. Each grid contains all rows. | The number of rows in the grid are determined by the Page Size. |
| Grid Rows is not 0 or the default value. Grid Cols is 0. | The table is broken into grids that are Grid Rows wide. Each grid contains all columns. | The number of columns in the grid are determined by the Page Size. |
| Grid Rows and Grid Cols are both not set to 0 or their default values. | The table is broken into grids of the requested size. | |
| Grid Rows and Grid Cols are both set to their default values. | The table is broken into grids of the default size. | |

Also note that once the grid size has been established for a sheet in a spreadsheet or database, the sheet cannot be re-exported with different grid dimensions. The sheet may be re-exported, however, if grids are disabled using sections.current.body.

Size calculations are performed using approximations. It is expected that each cell in the grid will contain roughly 10 characters of content. Therefore, the number of cells in the grid will be roughly the page size divided by 10. Setting the pageSize option will not cause content to be truncated if it exceeds the 10 characters of content expected in a given cell. Note that the pageSize option is never used to force a grid to break into pages. Thus, once the grid dimensions have been established, no page breaking is performed on the grid.

The default value for this option was chosen to prevent problems with large spreadsheets, which can consume conversion time while creating unmanageable output. Together with the default grid rows option value, the default grid dimensions represent the largest table size HTML Export can produce that will not result in browsers locking up when they try to read the file.

The solution to this large spreadsheet problem depends on whether or not page breaking is in effect:

- If page breaking is being used, use the maxreps attribute of the `{## repeat}` macro to prevent large files from becoming unmanageable.
- If page breaking is NOT being used, spreadsheets should be exported by inserting only the first grid of the spreadsheet (grids.1.body). Don't use a `{## repeat}` loop to get all the grids. Test for the existence of a second grid (grids.2.body). If this grid exists, then have the template write out a message indicating that the spreadsheet's contents were truncated on export.

Implementing support for spreadsheets in this manner rather than by inserting sections.current.body improves performance only when outputting very large spreadsheets. In these special cases, only the first grid is exported, resulting in significant performance savings. This savings also has the side benefit of producing an output file that most Web browsers will have little trouble displaying.

### Export Product

HTML Export

### Data Type

xsd:unsignedInt

### Data

Number of columns in the grid. Use "0" (zero) to not limit the number of columns in the grid.

### Default

- 100: This value is subject to change.

### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### gridRows

Options related to grids have no effect on the output unless a template that has been enabled with the `{## unit}` template macro is in use.

This option allows the developer to specify the number of rows that each template "grid" (applicable only to spreadsheet or database files) should contain. As such, it is only useful when a grid-enabled template has been selected with the template option.

Setting this option to zero ("0") means that no limit is placed on the number of rows in the grid. However, the settings of the pageSize, gridCols and gridRows options must all be taken into account when determining the actual dimensions of the grids used during an export.

Also note that once the grid size has been established for a sheet in a spreadsheet or database, the sheet cannot be re-exported with different grid dimensions. The sheet may be re-exported, however, if grids are disabled using sections.current.body.

Size calculations are performed using approximations. It is expected that each cell in the grid will contain roughly 10 characters of content. Therefore, the number of cells in the grid will be roughly the page size divided by 10. Setting the pageSize option will not cause content to be truncated if it exceeds the 10 characters of content expected in a given cell. Note that the pageSize option is never used to force a grid to break into pages. Thus, once the grid dimensions have been established, no page breaking is performed on the grid.

The default value for this option was chosen to prevent problems with large spreadsheets, which can consume conversion time while creating unmanageable output. Together with the default grid columns option value, the default grid dimensions represent the largest table size HTML Export can produce that will not result in browsers locking up when they try to read the file.

The solution to this large spreadsheet problem depends on whether or not page/deck breaking is in effect:

- If page breaking is being used, use the maxreps attribute of the `{## repeat}` macro to prevent large files from becoming unmanageable.
- If page breaking is NOT being used, spreadsheets should be exported by inserting only the first grid of the spreadsheet (grids.1.body). Don't use a `{## repeat}` loop to get all the grids. Test for the existence of a second grid (grids.2.body). If this grid exists, then have the template write out a message indicating that the spreadsheet's contents were truncated on export.

Implementing support for spreadsheets in this manner rather than by inserting sections.current.body improves performance only when inputting very large spreadsheets. In these special cases, only the first grid is exported, resulting in significant performance savings. This savings also has the side benefit of producing an output file that most Web browsers will have little trouble displaying.

#### Export Product

HTML Export

#### Data Type

xsd:unsignedInt

### Data

Number of rows in the grid. Use "0" (zero) to not limit the number of rows in the grid.

### Default

- 5000: This value is subject to change.

### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

## gridWrap

Options related to grids have no effect on the output unless a template that has been enabled with the `{## unit}` template macro is in use.

This option allows the developer to specify how the "previous" and "next" relationships will work between grids at the edges of the spreadsheet or database. As such, it is only useful when a grid-enabled template has been selected with the template option.

This option is best explained by example. Consider a spreadsheet that has been broken into 9 grids by HTML Export as follows:

Grid 1 Grid 2 Grid 3

Grid 4 Grid 5 Grid 6

Grid 7 Grid 8 Grid 9

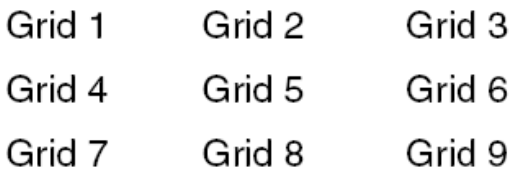
If this option is set to true, then the grids.next.body value after Grid 3 will be Grid 4. Likewise, the grids.previous.body value before Grid 4 will be Grid 3.

If this option is set to false, then the grids.next.body after Grid 3 will not exist as far as template navigation is concerned. Likewise, the grids.previous.body before Grid 4 will not exist as far as template navigation is concerned.

In other words, this option specifies whether the "previous" and "next" relationships "wrap" at the edges of the spreadsheet or database. As such, it is only useful when a grid-enabled template has been selected with the template option.

This option is best explained by example. Consider a spreadsheet that has been broken into 9 grids by HTML Export as follows:

***Figure 5 Spreadsheet Broken Into Grids***




If this option is set to true, then the grids.next.body value after Grid 3 will be Grid 4. Likewise, the grids.previous.body value before Grid 4 will be Grid 3.

If this option is set to false, then the grids.next.body after Grid 3 will not exist as far as template navigation is concerned. Likewise, the grids.previous.body before Grid 4 will not exist as far as template navigation is concerned.

In other words, this option specifies whether the “previous" and “next" relationships “wrap" at the edges of the spreadsheet or database.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

- true: To continue past the edge of the spreadsheet.
- false: To stop at the edge of the spreadsheet.

#### Default

true

#### Equivalents

C Client Implementation: XSD_boolean

JAVA Client Implementation: Boolean

### javaScriptTabs

Tab support is available by setting this option to true. When active, this option uses JavaScript to calculate tab stops and position blocks of text accordingly. Potential side effects of this include delays in loading the pages in the browser and seeing the text initially with no whitespace at all followed by a pause and then all of the tabs popping into place. In addition, this support is limited to only left tabs.

In order to take advantage of this option the following additional steps must be taken:

1. The template must contain a <script> tag. Something similar to the following code fragment is recommended:

```
{## if element=pragma.jsfile}
<script language="Javascript1.2" src="{## insert
element=pragma.jsfile}"></script>
{## /if}
```

2. The template must also run the DoTabStops routine in the <body> of the HTML. A span tag used to define the value of oneinch should follow this. Something similar to the following code snippet is recommended to accomplish this:

```
{## if element=pragma.jsfile}
   <body onload="DoTabStops()">
   <span id="oneinch" style="width: 1in"></span>
{## else}
   <body>
{## /if}
```

3. A flavor of HTML that supports CSS must be used.

4. The user's browser must support JavaScript and this support must be enabled.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true: Use JavaScript to create tabs.
- false: No tab support.

#### Default

- false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### pageSize

This option sets a suggested page size for the output generated. This means that the text of the document is broken up into "pages" of approximately the requested size. Each page is stored as a separate output file.

This feature is particularly useful when converting documents that are poorly structured. Many documents lack the kind of style information HTML Export normally uses to break the document into pieces based on things like headings. By setting this option, the exported document can be presented as a set of more manageable pieces rather than a single giant output file. It is also useful with documents that are structured but have large pieces in the structure.

If page breaking is activated (set to a non-zero value), HTML Export will buffer the entire output document in memory during conversion. Conversion times and memory requirements will increase accordingly in this case.

The size specified by this option is given in characters of text. Only text inserted from the input document is counted in the page size. Thus, "as is" text from the template is not counted against the page size. Also, markup tags are not counted in the page size. In addition, some template inserts are normally used as attributes to markup tags, and as such they are not counted in page size calculations no matter how they are actually used. Those template inserts are:

- pragma.charset
- pragma.jsfile
- pragma.cssfilename
- sections.x.itemnum
- sections.x.reflink

A page size of zero ("0") indicates that this option is turned off and no page breaking is done.

When this option is turned on, the page breaking rules are as follows:

- Hard page breaks in the document always trigger a page break. Soft page breaks are ignored.
- A page break may be specified in the template with the `{## unit break}` macro.
- A page boundary will never be created in the middle of a paragraph. As many paragraphs as possible will be written without exceeding the requested page size. Page sizes are not hard limits on content however. One situation where the page size could be exceeded would be if a single paragraph exceeds the page size.
- When grid-enabled templates are in use, the exported grids are not broken based on the setting of this option. However, this option may affect the size of grids generated.
- Use of this option will not cause the contents of cells within a grid to be truncated.
- When grids are not in effect, spreadsheets and databases will be broken based on page size. For these section types, checks for page breaks will be made after each full row from the spreadsheet or database is written.

It is up to the template author to then connect these pieces with the appropriate links. In order to use this option, the template must be equipped to use the `{## unit}` syntax.

Note that templates enabled with the `{## unit}` syntax may be mixed with templates that do not contain `{## unit}` macros. In this case, page breaking will only occur in the template that is enabled with `{## unit}` macros. An example of where this would be desirable is a "table of contents" template that uses two sub templates to each fill in the contents of a frame. The frame containing the actual table of contents could avoid being broken into pages by not containing any `{## unit}` macros. The frame containing the actual document contents could then support paging by using `{## unit}` macros.

#### Export Product

HTML Export

#### Data Type

xsd:unsignedInt

#### Data

Approximate page size in characters.

#### Default

- 0: No page size limit.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### preventGraphicOverlap

Most browsers support flowing text around images. Unfortunately, even the most popular browsers also have bugs in their support for this feature that occasionally result in document elements overlapping. This option allows users of HTML Export to choose if they would rather have text flowing around graphics or if they are willing to sacrifice that feature in order to prevent browser overlap bugs.

When this option is turned on (set to true), HTML Export prevents browsers from causing graphic overlap problems by surrounding all <img> tags with  tags. The overlap problems occur most frequently when the browser is displaying a document that has a mix of left- and right-aligned graphics in close proximity to each other. Resizing the browser window horizontally will sometimes expose this problem if it does not appear initially.

Because these browser bugs are infrequently seen, this option is turned off (set to false) by default. However, setting this option to false does not guarantee that text will be able to flow around graphics in the browser the same way it does in the original document. There are two problems which can prevent this from occurring.

The first problem is that when objects are placed using positional frames. Unfortunately, most new word processing formats do this automatically. When positional frames are used, each object exists in its own frame. HTML Export converts each frame as a single paragraph. Therefore, the objects are written one after the other even if they were originally placed side by side in the source document.

The second problem is associated with image alignment. For some images, HTML Export is unable to obtain the alignment of the image, so the alignment of the paragraph it is contained in is used instead. The reason HTML Export uses this alignment, which is not necessarily 100% correct, is because without adding "align=" in the <img> tag, text does not wrap around images in browsers.

Also note that setting this option to false will have no effect if the strictDTD flag of the compliance option is set. This is because  tags are not allowed inside <p> tags, so HTML Export cannot use  tags to prevent graphics from overlapping.

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Data

- true: Allow text flow around graphics.
- false: Prevent browser image overlap problems.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean

- JAVA Client Implementation: Boolean

### template

This option allows the developer to specify the template file that the technology uses to generate its output.

#### Export Product

HTML Export

#### Data Type

IOSpec

#### Data

A file location specification. The following are the currently valid IOSpec values:

- path: A file-system path to a file.
- url: A URL to a file.
- redirect: A spec that can be opened by an open function that is supplied by the caller. This type is only supported when the calling application is using the C/C++ or Java client API.

#### Default

If no template is specified, a standard template is used.

#### Equivalents

- C Client Implementation: TS_IOSpec
- JAVA Client Implementation: IOSpec

## Compression

This section discusses compression options.

### applyZLIB

This option determines if ZLIB compression will be applied to fonts and raster graphics while generating the PDF output file.

#### Export Product

PDF Export

#### Data Type

xsd:boolean

#### Data

- true: ZLIB compression is applied to fonts and embedded graphics.
- false: ZLIB compression is not applied to fonts and embedded graphics.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolen

### allowJPEG

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.

#### Export Product

HTML Export, Image Export, PDF Export, XML Export

#### Data Type

xsd:boolean

#### Data

- true: Allow access to files that use JPEG compression
- false: Do not allow access to files that use JPEG compression

#### Default

true

### allowLZW

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: TIF files will still be created when LZW compression is used and this option is enabled, but the resulting images will be large, uncompressed TIF files)
- PDF files that use internal LZW compression
- ZIP and self-extracting archive (.EXE) files containing "shrunk" files
- Postscript files using LZW compression

Image Export will not be affected by this option when processing formats that compress subfile contents but not subfile names, such as TAR and ZIP.

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.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:boolean

#### Data

- true: LZW compressed files will be read and written normally.
- false: LZW compressed files will not be read or written.

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## Graphics

This section discusses graphics options.

### acceptAlternateGraphics

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.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Data

- true: gif, jpeg, and png embeddings will be extracted, not converted. All other embeddings will be converted to the format specified by graphicType. If graphicType is set to 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 none, no embeddings will be extracted or converted.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### graphicGifInterlaced

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.

#### Export Product

HTML Export, Image Export, XML Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true: Produce interlaced GIFs.
- false: Produce non-interlaced GIFs.

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### graphicCropping

When set to cropToContent, this option forces Image Export to crop whitespace from the edge of each output image. This includes margins and any unused space at the end of a page. This results in smaller output files without any loss of original input document content.

If there is no content, then no cropping is performed.

#### Export Product

Image Export

#### Data Type

GraphicCroppingEnum

#### Data

- cropToContent: Files are cropped to smallest bounding rectangle.
- noCropping: Files are not cropped.

#### Default

noCropping

#### Equivalents

- C Client Implementation: OIT_GraphicCroppingEnum
- JAVA Client Implementation: GraphicCroppingEnum

### graphicHeight

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 graphicWidthLimit options can override the setting for graphicHeight.

#### Export Product

Image Export

#### Data Type

xsd:unsignedInt

#### Data

The desired absolute height of the output image, in pixels.

#### Default

- 0: No absolute height specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicHeightLimit

This option 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 of whether the graphicWidthLimit 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 height of graphics by using 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.

#### Export Product

HTML Export, Image Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The maximum height of the output graphic in pixels. A value of zero causes this option to be ignored.

#### Default

- 0: No absolute height limit specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicOutputDPI

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.

You may also specify the value 0 for the DPI, which will cause the output image to be created with identical pixel dimensions as the original input image, without consideration for physical measurements of image size.

Setting this option to 0 may result in the creation of extremely large images. Be aware that there may be limitations in the system running this technology that could result in undesirably large bandwidth consumption or an error message. Additionally, an out of memory error message will be generated if system memory is insufficient to handle a particularly large image.

Also note that the 0 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.

#### Export Product

HTML Export, Image Export, PDF Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The DPI to use when exporting graphic images. The maximum value allowed is 2400 DPI.

#### Default

- 96: 96 dots per inch.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicSizeLimit

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.

graphicSizeLimit 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.

#### Export Product

HTML Export, Image Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

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

#### Default

- 0: Option is turned off.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicSizeMethod

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.

#### Export Product

HTML Export, Image Export, PDF Export, XML Export

#### Data Type

GraphicSizeMethodEnum

#### Data

One of the following values:

- quick: Resize without antialiasing
- smooth: Resize using antialiasing
- smoothGray: Resize using antialiasing for grayscale graphics only (no antialiasing for color graphics)

#### Default

smooth

#### Equivalents

- C Client Implementation: OIT_GraphicSizeMethodEnum
- JAVA Client Implementation: GraphicSizeMethodEnum

### graphicTransparencyColor

This option allows the user to set the color used as the "transparency color" in the output graphic file. Naturally, this option is only used when the selected output graphic file format supports transparency (GIF and PNG only). If the option is not set, the default behavior is to use the same color value that the input file used as the transparency color.

Use the SCCVWRGB(r, g, b) macro to create the color value to pass to this option. The red, green and blue values are percentages of the color from 0-255 (with 255 being 100%). Note that this macro should be used to set a variable of type xsd:unsignedInt and that variable should then be passed to the set option routine (instead of trying to use the macro as part of the set option call directly).

Since there is no way to "unset" an option once it has been set, the developer may set the option to -1 if they wish to revert to the default behavior.

#### Export Product

HTML Export, Image Export

#### Data Type

xsd:unsignedInt

#### Data

An RGB color value created with the SCCVWRGB(r, g, b) macro.

#### Default

- -1: Use the same transparency color as the source document.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicType

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 noGraphics. Some of the Outside In Viewer Technology's import filters can be optimized to ignore certain types of graphics.

It should be noted that unpredictable and potentially undesirable output will occur if this option is set to noGraphics when a transformation is initiated and the template then attempts to use the `{## option}` macro to turn graphics back on.

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

#### Export Product

HTML Export, XML Export

#### Data Type

GraphicTypeEnum

#### Data

One of the following values:

- gif: GIF graphics
- jpeg: JPEG graphics
- png: PNG graphics
- noGraphics: Graphic conversion will be turned off

#### Default

jpeg

#### Equivalents

- C Client Implementation: OIT_GraphicTypeEnum
- JAVA Client Implementation: GraphicTypeEnum

### graphicWidth

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.

#### Export Product

Image Export

#### Data Type

xsd:unsignedInt

#### Data

The desired absolute width of the output image, in pixels.

#### Default

- 0: No absolute width specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicWidthLimit

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 of 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.

#### Export Product

HTML Export, Image Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The maximum width of the output graphic in pixels. A value of zero causes this option to be ignored.

#### Default

- 0: No absolute width limit specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### tiffOptions

This option allows the developer to specify sub-options unique to the TIFF output file type, in order to control color depth, byte structure and rendering method.

This option is only required if the output type is set to tiff.

When any of the CCITT compression models are used, the color space must be black and white (Black and White).

#### Export Product

Image Export

#### Data Type

This data type is a structure composed of values that set options specific to generating TIF images of input files. The structure is defined as follows:

```
<complexType name="tiffOptions">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="colorSpace" type="ts:TiffColorSpaceEnum" minOccurs="0"
maxOccurs="1"/>
    <element name="compression" type="ts:TiffCompressionEnum" minOccurs="0"
maxOccurs="1"/>
    <element name="byteOrder" type="ts:TiffByteOrderEnum" minOccurs="0"
maxOccurs="1"/>
    <element name="fillOrder" type="ts:TiffFillOrderEnum" minOccurs="0"
maxOccurs="1"/>
    <element name="createOneFile" type="xsd:boolean" minOccurs="0"
maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

### Parameters

TiffOptions is a complexType data structure composed of five elements, corresponding to the color space, compression, byte-ordering, fill-ordering, and multi-page aspects of generated TIFF files. The elements of this data type are as follows:

- colorSpace: This element has a data type of TiffColorSpaceEnum, which contains values that specify the TIFF color depth and color options. The following settings are valid:
  - Black and White: 1 bit black and white
  - 8 Bit Palette: 8 bit palette
  - 24 Bit RGB: 24 bit RGB (this is the default value for this parameter)
- compression: This element has a data type of TiffCompressionEnum, which contains values that specify the type of compression used in the TIFF file that is generated. The following settings are valid:
  - None: No compression
  - PackBits: Packbits compression (this is the default value for this parameter)
  - LZW Compression: LZW compression
  - CCITT - 1D: CCITT – 1D. Please note that when setting this type of compression, colorSpace must be set to Black and White.
  - CCITT - Group 3 Fax: CCITT – Group 3 Fax. Please note that when setting this type of compression, colorSpace must be set to Black and White.
  - CCITT - Group 4 Fax: CCITT – Group 4 Fax. Please note that when setting this type of compression, colorSpace must be set to Black and White.
- byteOrder: This element has a data type of TiffByteOrderEnum that determines the byte order used within the file:
  - tiff_big_endian: This will use "big-endian" (Motorola) byte ordering.
  - tiff_little_endian: This will use "little-endian" (Intel) byte ordering (this is the default value for this parameter).
- createOneFile: This element has an xsd:boolean data type that determines whether or not multi-page TIFFs will be created. These are additional flags for setting other options in a TIFF file:
  - false: A TIFF image will be created for each page of a multi-page input document. No flags are used (this is the default value for this parameter).
  - true: The output of multiple pages from one input document will generate a single file with a separate image for each page converted.
- fillOrder: This element has a data type of TiffFillOrderEnum that determines the fill order used within the file. The value of this element only affects TIFF output when the compression element is set to CCITT - Group 3 Fax and the colorSpace element is set to Black and White.
  - fillOrder-1: Selects TIFF fill order 1, which is the default for this parameter and is recommended for most TIFF files intended for file stores.

- fillOrder-2: Selects TIFF fill order 2, which is the recommended fill order for TIFF files intended for electronic transmission (for example, fax).

#### Equivalents

- C Client Implementation: OIT_TiffOptions
- JAVA Client Implementation: TiffOptions

### graphicJpegQuality

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

#### Export Product

HTML Export, Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### 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

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### 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.

#### Export Product

HTML Export, Image Export, PDF Export, XML Export

#### Data Type

xsd:Boolean

#### Default

false

#### Equivalents

C Client Implementation: XSD_boolean

Java Client Implementation: Boolean

### renderEmbeddedFonts

Specifies if the embedded fonts from the PDF input file should be used to render text.

#### Export Product

Image Export

#### Data Type

xsd:Boolean

#### Default

true

#### Equivalents

C Client Implementation: XSD_boolean

Java Client Implementation: Boolean

## Spreadsheet and Database File Rendering

This section discusses spreadsheet and database options.

### databaseFitToPage

This option scales a spreadsheet file to a certain percent or to a page width or height. However, in an effort to preserve readability after scaling, Image Export will not shrink a database document to under approximately one-third of its original size.

It should be noted that when this option is set to dbNoScaling, the pages of the database file are printed down first and then across.

Please note that any margins applied as a result of settings for the defaultMargins option will be included in any scaling that is applied to the output image as a result of settings for this option.

#### Export Product

Image Export, PDF Export

#### Data Type

DatabaseFitToPageEnum

#### Data

One of the following values:

- dbNoScaling: This will not do any scaling of the database image. It will render in its original size onto as many pages as are required to fit the data.

- dbFitToPages: This will fit the database to one page, scaling to the image width or height depending on the page size and database size.
- dbFitToWidth: This will scale the database on the rendered image so it is no larger than one page wide.
- dbFitToWidth: This will scale the database on the rendered image so it is no larger than one page high.

#### Default

dbFitToPages

#### Equivalents

- C Client Implementation: OIT_DatabaseFitToPageEnum.
- JAVA Client Implementation: DatabaseFitToPageEnum

### databaseShowGridLines

If this option is true, lines are generated between cells in the rendered images.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### databaseShowHeadings

If this option is true, field headings will be generated along with the data.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Default

true

### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## maxSsDbPageHeight

Normally, the size of images generated from spreadsheet worksheets and database tables is limited to the size of the page defined by the input document's page size information and how the useDocumentPageSettings, graphicWidth and graphicHeight options are set. If, after scaling is factored in, the resulting image is too large to fit on a single page, it is split up into multiple pages.

The maxSsDbPageWidth and maxSsDbPageHeight options are used to change the size of a page to match the scaled size of the page being rendered - within limits. The key reason for those limits is that rendering very large pages can easily overwhelm the memory available on the system. When using this feature, a calculation should be made to be sure that the values passed in work within said memory limits. The values for these two options will override the current page dimensions if necessary.

The memory needed may be calculated based on the following:

```
memory = [max. worksheet/table height (in inches)]  x  [max. worksheet/table
width (in inches)]  x  [dpi setting]2  x  3 bytes/pixel  +  a bit extra for the
needs of the rest of the conversion
```

By default, these options are set to the current page dimensions. Users may choose to set only one of the two options if desired. If, for example, only the maxSsDbPageWidth is set, then the height of the page will be based on the normal page height.

When a worksheet or table is larger than the maximum values specified by these options, then the file is rendered on multiple pages, with the requested (larger) page dimensions.

These new options grow the page size (if needed) to match the size of the worksheet or table. This differs from the graphicWidth and graphicHeight options, which set an absolute page size without regard to the size of the worksheet or table.

If text in cells ends up extending past the edge of the cell and beyond the edge of the page, Image Export writes one or more additional pages for the overflow text.

### Export Product

Image Export, PDF Export

### Data Type

xsd:unsignedInt

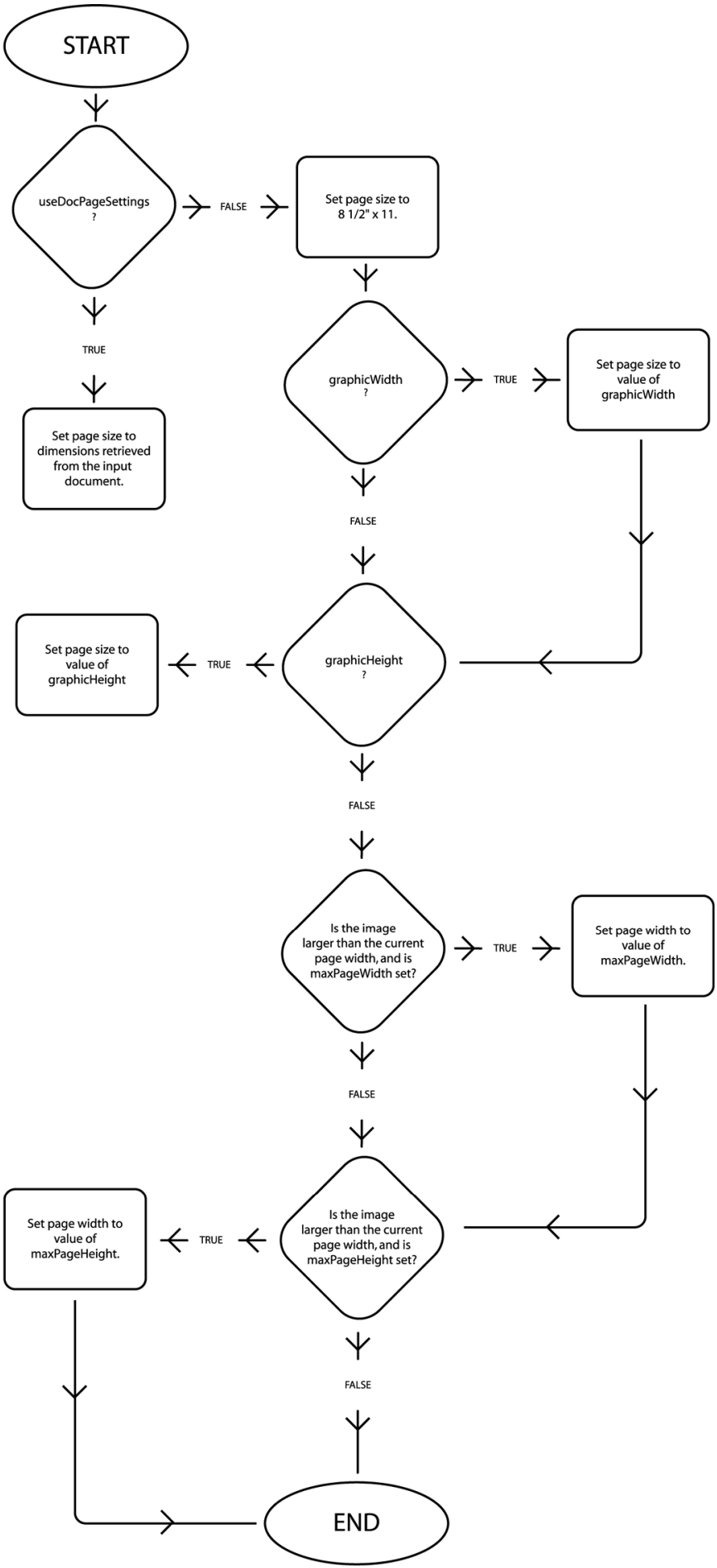
### Data

The maximum page height (including margins) specified in twips (1440 twips are in 1 inch). If the value specified is smaller than the page height, then Image Export will return a SCCERR_INVALIDMAXSSDBPAGE error.

### Default

- 0: Use the page height defined by the input document's page size information and by the useDocumentPageSettings, graphicHeightLimit and graphicHeight options.

***Figure 6 Logic Flow for Determining Page Size of Spreadsheet and Database Pages***



### maxSsDbPageWidth

See the documentation for maxSsDbPageHeight for a full discussion of how this option works and interacts with other options affecting the page size of images generated from spreadsheet and database pages.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Data

The maximum page width (including margins) specified in twips (1440 twips are in 1 inch). If the value specified is smaller than the page width, then Image Export will return a SCCERR_INVALIDMAXSSDBPAGE error.

#### Default

- 0: Use the page width defined by the input document's page size information and by the useDocumentPageSettings, graphicWidthLimit and graphicWidth options.

### showSpreadsheetBorder

This option has been deprecated beginning with the 8.2 version of the product. Please use the spreadsheetShowHeadings and spreadsheetBorders options instead.

This option affects database files the same way it affects spreadsheets.

This option allows users to speed up the conversion of large, sparse spreadsheets by turning off the table borders HTML Export generates by default (true is the default setting for this option). Setting this option to false turns off table border generations, reducing the amount of HTML written and enabling rowspan and colspan table tag attributes so that empty cells can be skipped. For large, mostly empty spreadsheets, this can result in greatly reduced conversion time and output file size(s). The output appears in a format similar to that used by the original application when printing the file.

The default is to show borders (option set to true). This prevents problems with most browsers, which tend to render the text in a way that makes adjacent cells hard to distinguish. This output appears in a browser in a format similar to that used by the original application when displaying the file on-screen.

This option must be set to the default value when the output format does not support tables.

When the option is set to false, the following caveats apply:

- If the spreadsheet being processed stores data by row (such as Microsoft Excel spreadsheets) rather than by column (such as Quattro files), additional optimizations are possible. The technology will use colspan to shrink the output when two or more adjacent cells in a row are empty. When two or more adjacent rows are completely empty, they are ignored and not included in the output.

- Note that if there are merged cells in the input document, the technology will not produce perfectly optimized output. Instead, rowspan and colspan will not be used to compress empty cells until after the merged cells are processed.
- This option disables the creation of Row and Column headings ("1", "2", "3" / "A", "B", "C").

#### Export Product

HTML Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### spreadsheetBorders

This option supersedes some of the functionality from the now discontinued showSpreadsheetBorder option.

This option determines how borders will be handled for spreadsheet and database files.

There are three valid values for this option:

- createBorderIfMissing: If a CSS output flavor is in use, this forces borders to be created if none are present in the (entire) table. By default, most apps do not include borders when creating these types of files. When needed, HTML Export will generate thin borders between cells. Otherwise, the borders specified in the table are used.

  Using borders makes it easier to read the output data by preventing values from running together when there is not much space between cells. This output appears in a browser in a format similar to that used by the original application when displaying the file on-screen.

  The behavior of this setting matches the old default border behavior of the discontinued showSpreadsheetBorder option.

  If a CSS output flavor is not in use, then borders are put around all cells no matter how the input document is formatted. This is because individual cell border information may not be specified in HTML without CSS.

  This is the default behavior for this option.

- bordersOff: This setting forces the borders always to be off, regardless of borders specified in the source document. This option setting does not distinguish between CSS and non-CSS output flavors being used. Turning borders off has the following advantages:

- It allows HTML Export to use optimizations that speed up the conversion of large, sparse files. It does this by enabling rowspan and colspan table tag attributes to be used to span empty cells. It also reduces the amount of HTML needed to be written for individual cells. For large, mostly empty spreadsheets, this can result in greatly reduced conversion time and output file size(s). The output appears in a format similar to that used by the original application when printing the file.
- For left aligned text and data cells, a special optimization has been made to merge those cells with any empty cells on the right.

The following caveats apply to the optimization:

- If the spreadsheet being processed stores data by row (such as Microsoft Excel spreadsheets with portrait page orientation) rather than by column (such as Quattro files), additional optimizations are possible. The technology will use colspan to shrink the output when two or more adjacent cells in a row are empty. When two or more adjacent rows are completely empty, the rows are skipped, and the row height of the next non-empty row is increased.
- Note that if there are merged cells in the input document, the technology will not produce perfectly optimized output. Instead, colspan will not be used to compress empty cells until after the merged cells are processed.
- The behavior of this option setting matches the old border behavior of the now discontinued showSpreadsheetBorder option when it was set to FALSE. However, this option does not disable the creation of Row and Column headings ("1", "2", "3" / "A", "B", "C"). To do that, use the new spreadsheetShowHeadings option.
- If the current row has frames in it, we will not span those cells.

- useSourceBorders: If a CSS output flavor is being used, then this value sets the borders according to what is specified in the source document.

  If a CSS output flavor is not in use, then borders are put around all cells no matter how the input document is formatted. This is because individual cell border information may not be specified in HTML without CSS.

### Export Product

HTML Export

### Data Type

SpreadSheetBordersEnum

### Data

- createBorderIfMissing: Use source document borders. If no borders are in the table, automatically create borders.
- bordersOff: Do not write any table borders.
- useSourceBorders: Use source document borders.

#### Default

- createBorderIfMissing

### showHiddenSpreadsheetCells

This option lets you determine whether or not to show hidden rows or columns when rendering spreadsheets. It is used to expand the widths of cells that are hidden by virtue of having their row height or column width reduced to 0. This is a BOOLEAN option that will leave the data hidden when it is false, and show all hidden rows and columns when it is true, displayed using the default row width or default column height.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Data

- true: Displays hidden cells.
- false: Does not display hidden cells.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### spreadsheetPageDirection

This option controls the pattern in which the pages are rendered, either across first and then down, or down first and then across.

This option is overridden when the useDocumentPageSettings option is set to true and print direction is specified in the input document.

#### Export Product

Image Export, PDF Export

#### Data Type

SpreadSheetPageDirectionEnum

#### Data

One of the following values:

- acrossThenDown: Will specify that pages are printed across first and then down.

- downThenAcross: Will specify that pages are printed down first and then across.

#### Default

downThenAcross

#### Equivalents

- C Client Implementation: OIT_SpreadsheetPageDirectionEnum
- JAVA Client Implementation: SpreadsheetPageDirectionEnum

### spreadsheetFitToPage

This option requests that the spreadsheet file be fit to one page.

Please note that any margins applied as a result of settings for the defaultMargins option will be included in any scaling that is applied to the output image as a result of settings for this option.

This option is overridden when the useDocumentPageSettings option is set to true and fitting the page to the printer's image limits is specified in the input document.

#### Export Product

Image Export, PDF Export

#### Data Type

SpreadsheetFitToPageEnum

#### Data

One of the following values:

- ssNoScaling: No scaling is performed on the spreadsheet image. It will render in its original size onto as many pages as are required to fit the data.
- ssFitToPages: Will scale the spreadsheet in the rendered image to fit to the number of pages specified in the spreadsheetScaleXPagesHigh and spreadsheetScaleXPagesWide options. Since aspect ratio is maintained, the lesser of the two dimensions (width or height) will determine the scale factor. Note that if either spreadsheetScaleXPagesHigh or spreadsheetScaleXPagesWide is set to 0, the value in the other option will be nullified.
- ssFitToWidth: Will scale the spreadsheet in the rendered image so it is no larger than one page wide.
- ssFitToHeight: Will scale the spreadsheet in the rendered image so it is no larger than one page high.
- ssScaleByPercentage: Will scale the spreadsheet in the rendered image using the scale value stored in the spreadsheetScalePercentage option.

#### Default

- ssScaleByPercentage: Scales the rendered image of the spreadsheet using the scale value stored in the spreadsheetScalePercentage option (which is 100 by default).

#### Equivalents

- C Client Implementation: OIT_SpreadsheetFitToPageEnum
- JAVA Client Implementation: SpreadsheetFitToPageEnum

### spreadsheetShowGridLines

If this option is true, a line is generated between cells in the rendered image.

This option is overridden when the useDocumentPageSettings option is set to true and printing grid lines between cells is specified in the input document.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### spreadsheetShowHeadings

If this option is true, row and column headings will be rendered along with the data.

This option is overridden when the useDocumentPageSettings option is set to true and printing column and row headers is specified in the input document.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### spreadsheetScalePercentage

This option will scale spreadsheet pages by the percentage specified. The option has no effect unless the spreadsheetFitToPage option is set to ssScaleByPercentage.

This option must take a value between 1 and 100. If any value outside of this range is used, the option will be ignored.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Default

100

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### spreadsheetScaleXPagesHigh

This option will fit the spreadsheet image to the number of vertical pages specified. The setting for this option will have no effect unless the spreadsheetFitToPage option is set to ssFitToPages.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Default

1

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### spreadsheetScaleXPagesWide

This option will fit the spreadsheet image to the number of horizontal pages specified. The setting for this option will have no effect unless the spreadsheetFitToPage option is set to ssFitToPages.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Default

1

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### generateExcelRevisions

If this option is true, the Excel spreadsheet revision data will be output in addition to the spreadsheet data itself.

#### Export Product

HTML Export, Search Export

#### Data Type

xsd:Boolean

#### Default

false

#### Equivalents

C Client Implementation: XSD_boolean

Java Client Implementation: Boolean

### omitEmptyEdgeCells

If true, this option will tell the spreadsheet filter to find the minimum rectangle that encloses all spreadsheet cells containing data. Any empty cells laying outside that rectangle will be omitted from the export. This does not eliminate all empty cells, only those cells on the edges, outside the calculated rectangle.

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:Boolean

#### Default

false

#### Equivalents

C Client Implementation: XSD_boolean

Java Client Implementation: Boolean

## Page Rendering

This section discusses age rendering options.

### defaultMargins

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

Please note all margins are applied before scaling with the databaseFitToPage, spreadsheetFitToPage, graphicHeight, graphicWidth, or graphicSizeLimit options.

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

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

#### Export Product

Image Export, PDF Export

#### Data Type

The defaultMargins structure is defined as follows:

```
<complexType name="defaultMargins">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="top" type="xsd:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <element name="bottom" type="xsd:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <element name="left" type="xsd:unsignedInt" minOccurs="0" maxOccurs="1"/>
    <element name="right" type="xsd:unsignedInt" minOccurs="0" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

#### Data

- top: Margin from the top edge of the image (in twips). Default is 1 inch.
- bottom: Margin from the bottom edge of the image (in twips). Default is 1 inch.
- left: Margin from the left edge of the image (in twips). Default is 1 inch.
- right: Margin from the right edge of the image (in twips). Default is 1 inch.

#### Equivalents

- C Client Implementation: OIT_DefaultMargins
- JAVA Client Implementation: DefaultMargins

### defaultPageHeight

This option sets the default page height of the output. The units for this value is determined by the defaultPageUnits option.

This is a floating point value.

#### Export Product

PDF Export

#### Data Type

xsd:float

#### Default

11 inches

#### Equivalents

- C Client Implementation: XSD_float
- JAVA Client Implementation: Float

### defaultPageWidth

This option sets the default page width of the output. The units for this value is determined by the defaultPageUnits option.

This is a floating point value.

#### Export Product

PDF Export

#### Data Type

xsd:float

#### Default

8.5 inches

#### Equivalents

- C Client Implementation: XSD_float
- JAVA Client Implementation: Float

### defaultPageUnits

This option allows the OEM to specify the units of measures for the defaultPageHeight and defaultPageWidth options.

If the developer changes the unit of measure, after having earlier set the units, then no attempt will be made to convert the height and width from the previous unit of measure to the current unit of measure.

One of the following:

- inches
- points

- centimeters
- picas

#### Export Product

PDF Export

#### Data Type

DefaultPageUnitsEnum

#### Default

inches

#### Equivalents

- C Client Implementation: OIT_DefaultPageUnitsEnum
- JAVA Client Implementation: DefaultPageUnitsEnum

### emailHeaderOutput

This option controls display of email headers in the output.

#### Export Product

Image Export, PDF Export

#### Data Type

EmailHeaderOutputEnum

#### Data

One of these values:

- emailHeaderStandard: Displays "To," "From," "Subject," "CC," "BCC," "Date Sent," and "Attachments" header fields only. The filter outputs any fields not listed above as hidden fields, so they will not display.
- emailHeaderAll: Displays all available email headers.
- emailHeaderNone: Displays no email headers.

#### Default

emailHeaderStandard

#### Equivalents

- C Client Implementation: OIT_EmailHeaderOutputEnum
- JAVA Client Implementation: EmailHeaderOutputEnum

### endPage

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

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

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Default

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

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### startPage

This option indicates the page rendering should start on. It is only valid if the option usePageRange has the value true.

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

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:unsignedInt

#### Default

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

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### useDocumentPageSettings

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

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

The values of the defaultMargins, spreadsheetShowGridLines, spreadsheetShowHeadings, spreadsheetPageDirection, and spreadsheetFitToPage options are overridden if this option is set to true and the properties associated with those options are specified in the input document. Additionally, print area and page breaks in spreadsheet documents are ignored unless this option is set to true.

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

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

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### usePageRange

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

#### Export Product

Image Export, PDF Export

#### Data Type

xsd:boolean

#### Data

One of the following values:

- true: The pages in the one-based, inclusive range from startPage to endPage will be printed.
- false: The entire document will be printed.

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## Font Rendering

This section pertains to font rendering options.

### defaultFont

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

#### Export Product

HTML Export, Image Export, PDF Export

#### Data Type

The defaultFont structure is defined as follows:

```
<complexType name="defaultFont">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="fontName" type="xsd:string" minOccurs="0" maxOccurs="1"
nillable="true"/>
    <element name="height" type="xsd:unsignedShort" minOccurs="0"
maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

#### Parameters

The DefaultFont option is a complexType data structure composed of two elements. The elements are as follows:

- fontName: An xsd:string value indicating the name of the font. For example, "Helvetica Compressed." The default is "Arial", however this default is constrained by the fonts available on the system.
- height: An xsd:unsignedShort value indicating the size of the font in half points. For example, a value of 24 will produce a 12-point font. This size is only applied when the font size is not known. The default is 10-point, however this default is constrained by the font sizes available on the system.

#### Equivalents

- C Client Implementation: OIT_DefaultFont
- JAVA Client Implementation: DefaultFont

### embedFonts

This option allows the developer to specify whether or not fonts should be embedded in the file. Note that for PDF/A compliance, this option must be set to true.

#### Export Product

PDF Export

#### Data Type

xsd:bool

#### Data

A Boolean value indicating if fonts should be embedded.

#### Default Value

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### excludeFont

This option allows the developer to specify a list of fonts that should be the only fonts excluded from the export process. To build a list of excluded fonts, the developer must issue an excludeFont option per font to be excluded.

Setting excludeFont followed by an includeFont will cause all the excludeFont to be ignored.

#### Export Product

PDF Export

#### Data Type

xsd:string

#### Data

The font to exclude.

#### Default

None.

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation: String

### includeFont

This option allows the developer to specify a list of fonts that should be the only fonts invluded in the export process. To build a list of included fonts, the developer must issue an includeFont option per font to be excluded.

Setting includeFont followed by an excludeFont will cause all the includeFont to be ignored.

#### Export Product

PDF Export

#### Data Type

xsd:string

#### Data

The font to include.

#### Default

None

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation: String

### fontDirectory

This option allows the developer to specify a directory where fonts are located that are valid for use by PDF Export.

This option must be set prior to performing any exports. Please note that PDF Export only supports *.ttf and *.ttc (TrueType) fonts, not *.fon (non-TrueType) fonts. Also, PDF Export does not require case-sensitive font filenames on UNIX systems.

#### Export Product

PDF Export

#### Data Type

xsd:string

#### Data

A path to the fonts.

#### Default

NONE - the option must be set.

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation: String

### fontAlias

This option sets or gets printer font aliases. For example, Chicago=Arial forces Chicago to be rendered as Arial.

#### Export Product

HTML Export, Image Export, PDF Export

#### Data Type

xsd:string

#### Data

The xsd:string value takes the form of font=alias, as in this example:

```
Chicago=Arial
```

The technology assumes the following default mappings. The first value is the font name, the second is the alias name.

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

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation:String

### strokeText

This option is used to stroke out (display as graphical primitives) text in an AutoCAD file. Setting this option to FALSE would improve performance, but the visual fidelity may be compromised.

- If the export for the conversion is text only, text is never stroked out.
- If the export is not text only, and the drawing is perspective, text will always be stroked out (regardless of this option). This is due to the fact that in non-text only situations visual fidelity is of importance, and handling of textual objects in perspective drawings is more accurate with stroked out text. If the conversion is non-text only and the drawing is not perspective, this option determines if text should be stroked.

Note that when this option is TRUE, some special characters appear as asterisks or question marks due to limited support of characters for stroking out text.

#### Data Type

xsd:bool

#### Default

TRUE

## Watermarks

This section discusses watermark options.

By default, the watermark image is centered in the middle of the target image.

### graphicWatermarkOpacity

This option must be set and defined to turn on watermarking support. A value of 0 is default and turns watermarking off. Values (1 to 255) specify a level of transparency. 255 is fully opaque. 1 is very transparent.

#### Export Product

Image Export

#### Data Type

xsd:unsignedInt

#### Data

A value between 1 and 255. The value of 0 turns watermarking off.

#### Default

0

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicWatermarkPath

This option needs to be set to specify a watermark file and path.

#### Export Product

Image Export

#### Data Type

xsd:string

#### Data

A pointer to a buffer containing a null-terminated string. The buffer must contain a path to the file containing the watermark image. The buffer can be no larger than sizeof(IMGWATERMARKPATH) bytes.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicWatermarkScaleType

Indicates whether to scale the watermark image or not. A value of scaleWatermarkOff means that we blend the watermark onto the original graphic with the original watermark height and width. This is the default value. A value of scaleWatermarkByPercent means that we will scale the watermark to be a certain percentage of the original watermark's height and width.

#### Export Product

Image Export

#### Data Type

GraphicWatermarkScaleTypeEnum

#### Data

- scaleWatermarkOff: When set means no scaling of the watermark image is to be done.
- scaleWatermarkByPercent: When set means that the watermark image is to be scaled to a percentage of its size. The percentage that is used is set by the graphicWatermarkScalePercent option.

#### Default

scaleWatermarkOff

#### Equivalents

- C Client Implementation: OIT_GraphicWatermarkScaleTypeEnum
- JAVA Client Implementation: GraphicWatermarkScaleTypeEnum

### graphicWatermarkScalePercent

Active when graphicWatermarkScaleType is set to scaleWatermarkByPercent. Values (1 to 100) scale the watermark to be a specified percent of its original size. A value of 100 (default) overlays the target image with the watermark image at its original size; e.g., if the original graphic watermark is 4x4 and the target image is 6x8, the graphic watermark will be scaled to 4x4 to overlay the target image.

#### Export Product

Image Export

#### Data Type

xsd:unsignedInt

#### Data

Values of 1 to 100 scale the watermark image to a percentage of the watermark image's size.

#### Default

100

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### graphicWatermarkHorizPos

Offset in pixels within target image to adjust watermark position. Default value is 0.

#### Export Product

Image Export

#### Data Type

xsd:int

#### Data

The number of pixels to offset the image in the horizontal direction. The watermark image is centered in the middle of target image, and then the watermark image is moved by this many pixels. A positive value moves the watermark towards the right, a negative value moves the watermark towards the left.

#### Default

0

#### Equivalents

- C Client Implementation: XSD_int

- JAVA Client Implementation: Integer

### graphicWatermarkVertPos

Offset in pixels to adjust the watermark position within the target image. Default value is 0.

#### Export Product

Image Export

#### Data Type

xsd:int

#### Data

The number of pixels to offset the image in the vertical direction. The watermark image is centered in the middle of the target image, and then the watermark image is moved by this many pixels. A positive value moves the watermark towards the bottom, and a negative value moves the watermark towards the top.

#### Default

0

#### Equivalents

- C Client Implementation: XSD_int
- JAVA Client Implementation: Integer

### enableWatermark

This option allows the developer to specify if a watermark should be included on each of the rendered PDF pages.

#### Export Product

PDF Export

#### Data Type

xsd:boolean

#### Data

Boolean indicating if a watermark is to be included in the rendering.

#### Default Value

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### watermarkHorizOffset

This option sets the horizontal offset of a watermark image relative to the center of a page. It is intended to be used when the watermarkPosition option is set to a value of offsetFromCenter. It should also be paired with the watermarkVertOffset option.

This value is in twips. 1440 twips equals 1 inch.

#### Export Product

PDF Export

#### Data Type

xsd:signedInt

#### Default

- 0: No offset specified.

#### Equivalents

- C Client Implementation: XSD_signedInt
- JAVA Client Implementation: SignedInt

### watermarkImage

This option specifies the location of the watermark image using an IOSpec. Redirected IO is allowed.

#### Export Product

PDF Export

#### Data Type

IOSpec

#### Default

No location specified.

#### Equivalents

- C Client Implementation: TS_IOSpec
- JAVA Client Implementation: IOSpec

### watermarkPosition

This option allows the developer to specify where on the page the watermark should be placed.

No scaling of watermark graphics is performed, so a graphic larger than the page or located such that the entire image cannot fit on the page will be cropped to fit the page.

#### Export Product

PDF Export

#### Data Type

WatermarkPositionEnum

#### Data

Enumeration containing a positional indicator for the watermark, as well as watermarkXOffset and watermarkYOffset values that are used.

#### Default Value

offsetFromCenter

#### Equivalents

- C Client Implementation: OIT_WatermarkPositionEnum
- JAVA Client Implementation: WatermarkPositionEnum

### watermarkScalePercent

This option specifies a fixed scaling amount for a watermark image. This option must be set in conjunction with watermarkScaling.

#### Export Product

PDF Export

#### Data Type

xsd:unsignedInt

#### Default

- 0: No scaling specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### watermarkScaling

This option specifies how scaling of the watermark is to be done. If the watermark is to be scaled by a fixed percentage, the pdfScale value must be used, and the watermarkScalePercent option must also be set.

#### Export Product

PDF Export

#### Data Type

WatermarkScalingEnum

#### Default

- 0: No scaling specified.

#### Equivalents

- C Client Implementation: OIT_WatermarkScalingEnum
- JAVA Client Implementation: WatermarkScalingEnum

### watermarkVertOffset

This option sets the vertical offset of a watermark image relative to the center of a page. It is intended to be used when the watermarkPosition option is set to a value of offsetFromCenter. It should also be paired with the watermarkHorizOffset option.

This value is in twips. 1440 twips equals 1 inch.

#### Export Product

PDF Export

#### Data Type

xsd:signedInt

#### Default

- 0: No offset specified.

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

## XML

This section pertains to XML options.

### allCapsOn

When set, causes capitalized text to be output and appropriately marked. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### boldOn

When set, causes bold text to be output and appropriately marked. Not valid for the SearchText and PageML output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### cellInfoOn

When set, SearchML will output a <cell> element that will encapsulate data from each non-empty cell in a spreadsheet. (Note: Numeric cells are considered empty unless FI DOCS NO HP BUILDING(3.7) is enabled. ) The <cell> element will have a required attribute start which will give the location of the cell. It will also have an optional attribute end which will be used to indicate a merged cell. Both the start and end attributes will be in the form RowColumn where the Row will be a letter and Column will be a number (for example, <cell start="A1">). Valid only for the SearchML 3.x output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### changeNumbertoTextOn

Includes data not originally stored as text in the input document. This can be important content the user would see when viewing the document in the original application (time and owner information in archives, numbers in spreadsheets/ databases, etc.). Valid for all output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### documentPropertiesOn

When set, document properties are included inthe output. Default value is false. Not valid for the PageML output format.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### doubleUnderlineOn

When set, causes double-underlined text to be included in the output and appropriately marked. Not valid for the SearchText and PageML output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### embeddingsOn

Include embeddings. Not valid for the PageML output format.

#### Export Product

Search Export,

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### errorInfoOn

When this flag is set, SearchML will output an <error> element if an error occurs while processing the main document or any sub-documents. The <error> element has one required attribute, code, which will be a hex value of the error code. The contents of the element will be a string with the description of the error returned from DAGetErrorString. Valid only for the SearchML 3.x output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean

- JAVA Client Implementation: Boolean

### generateSystemMetaDataOn

When this flag is set, system metadata will be generated. This text is "generated" and is part of the document properties. This information is gathered through system calls and may adversely affect performance. Valid only for the SearchML 3.x output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### hiddenCellInfoOn

When true, information regarding hidden cells will be output. This option is only relevant when cellInfoOn is also true.

#### Export Product

Search Export

#### Data Type

xsd:Boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- Java Client Implementation: Boolean

### hiddenOn

Include hidden text in the output. Not valid for the PageML output format.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### italicOn

Include italic text in the output. Not valid for the SearchText and PageML output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### metadataOnlyOn

Produce only metadata. Not valid for the PageML output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### originalCharsetOn

When this option is set, an attribute named oce is added either to <p> or <r> elements as appropriate. The value of the attribute is a hex representation of the character set. The value is defined by our core technology, SO_ANSIUNKNOWN for instance. Possible values for this attribute appear in the vtchars.h header file. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### outlineOn

Include outlined text in the output. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### produceURLsOn

Produce URL information when it is available. Valid for the SearchML 3.x and SearchHTML output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### revisionAddOn

When set, causes added text to be output and appropriately marked. Valid for all output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### revisionDeleteOn

When set, causes deleted text to be output and appropriately marked. Valid for all output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### revisionsOn

When set, revised or annotated text will be designated as such. Valid only for the SearchML 3.x output format.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### smallCapsOn

When set, causes text in small caps to be output and appropriately marked. Valid for the SearchML output format only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### strikeoutOn

When set, causes strikeout text to be output and appropriately marked. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### underlineOn

When set, causes underlined text to be output and appropriately marked. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### xmlDefinitionMethod

This option determines whether Search Export will reference a SearchML or PageML schema, DTD, or no reference when generating output. This option is not valid when SearchText or SearchHTML is the output format.

#### Export Product

Search Export, XML Export

#### Data Type

XMLDefinitionMethodEnum

#### Data

One of the following values:

- dtd: Document Type Definition (DTD)
- xsd: Extensible Schema Definition
- noDefinition: No XML definition reference

#### Default

noDefinition

#### Equivalents

- C Client Implementation: OIT_XmlDefinitionMethodEnum
- JAVA Client Implementation: XmlDefinitionMethodEnum

### xmlDefinitionLocation

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

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

#### Export Product

Search Export, XML Export

#### Data Type

xsd:string

#### Data

A UTF-8 encoded string specifying the location of an xsd or dtd file. If using the C API, this string must be a null-terminated array of single-byte characters.

#### Default

None

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation: String

### nullReplacementCharacter

This option specifies a two-byte Unicode character that will be used to replace null characters if null path separators are being used. This option defaults to '/' and is valid for the SearchML 3.x, SearchHTML and SearchText output formats.

#### Export Product

Search Export

#### Data Type

xsd:unsignedShort

#### Data

A two-byte Unicode character that will be used to replace null characters if null path separators are being used.

#### Default

0x002f = "/"

#### Equivalents

- C Client Implementation: XSD_unsignedShort
- JAVA Client Implementation: UnsignedShort

### printerName

This option is Windows-specific. It is used to set which device context to use to render the pages.

It specifies, as a byte string, the name of the printer whose metrics should be used to calculate pagination information. If unspecified, the default printer will be used. The screen metrics of the system will be used if a printer is not specified and a default printer does not exist. As pagination is affected by the metrics of the device context and installed fonts, PageML XML output can vary between different systems and configurations.

#### Export Product

Search Export

#### Data Type

xsd:string

#### Data

A null-terminated single-byte string for the name of the printer which is the device context that should be used to render pages.

#### Default

- null: PageML uses the Windows default printer.

#### Equivalents

- C Client Implementation: XSD_string
- JAVA Client Implementation: String

### paragraphStyleNamesOn

Include paragraph style name references as an attribute of paragraph tags. Valid for the SearchML 3.x output formats only.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### includeTextOffsets

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

#### Export Product

Search Export, XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### paragraphAttributes

This option allows the developer to track paragraph attributes contained in the input document and, optionally, include them in the XML output. All lengths are measured in twips. The values that appear in the SearchML output are the values that apply to the first content encountered in a given paragraph. For example, if the character height changes after the initial content in a paragraph, that change will be ignored. Left and first line indents are measured relative to the left page margin. The right indent is measured relative to the right page margin.

#### Export Product

Search Export

#### Data Type

The ParagraphAttributes data type is a structure composed of Boolean values that act as flags, determining what (if any) paragraph attributes should be included in SearchML output. The structure is defined as follows:

```
<complexType name="ParagraphAttributes">
 <complexContent>
  <extension base="xsd:anyType">
   <sequence>
    <element name="spacing" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
    <element name="height" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
```

```
    <element name="leftIndent" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
    <element name="rightIndent" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
    <element name="firstIndent" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

#### Data

The paragraphAttributes option is a complexType data structure composed of Boolean variables, which may be switched on or off in any combination. The variables are:

- spacing
- height
- leftIndent
- rightIndent
- firstIndent

#### Default

- 0: All flags set to false.

#### Equivalents

- C Client Implementation: OIT_ParagraphAttributes
- JAVA Client Implementation: ParagraphAttributes

### unmappedText

This option allows for the production of unmapped text (the original code points from the input document). A new <unmapped> element will be produced to enclose this text. The <unmapped> element will contain base64-encoded text. It will also contain two attributes. "OCE" will contain a hex value representing the character set. "Font" will contain a string value of the original font name. This is necessary for non-standard encodings such as wingdings or webdings. This option is only valid in the SearchML 3.2 (and higher) schema.

#### Export Product

Search Export

#### Data Type

SearchMLUnmappedTextEnum

#### Data

One of the following values:

- justUnmappedText: Output just the unmapped text
- noUnmappedText: Don't output any unmapped text.
- bothUnmappedText: Output both the original and the unmapped text.

#### Default

- noUnmappedText

#### Equivalents

- C Client Implementation: OIT_SearchMLUnmappedTextEnum
- JAVA Client Implementation: SearchMLUnmappedTextEnum

### suppressArchiveSubDocsOn

Subdocuments in archives are not processed. Not valid for the PageML output format.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### suppressAttachmentsOn

Attachments are not processed. Not valid for the PageML output format.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### textOutOn

This option is valid only for the PageML output format.

When set to true, include text in the PageML output.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### xmlDeclarationOff

Exclude the XML declaration. Not valid for the SearchText and SearchHTML output formats.

#### Export Product

Search Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### optimizeSections

When set to true, uses wp.section elements to delineate column references.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### charMappingDefault

When set to true, all text is mapped to Unicode in tx.text elements.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### charMappingNone

When set to true, all text is left in the original character set in tx.utext elements.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### charMappingText

When set to true, text is mapped to Unicode where possible, while unmappable text is left in the original character set.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### charMappingBoth

When set to true, both mapped and unmapped text are included as alt elements containing tx.text and tx.utext.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

`false`

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### convertChartObjects

The value of this option is a Boolean that when set to true forces all DateTime properties to be converted to the ISO 8601 standard. This conversion can only be performed using dates that are stored as numeric data within the original file. If the option is set to false, DateTime properties will not be converted.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean

- JAVA Client Implementation: Boolean

### convertDateTimeProperties

The value of this option is a Boolean that if set to true will convert ch.chart objects to a graphic file and the resulting file will be referenced by the locator child of the ch.chart. If the option is set to false, ch.chart objects will not be converted.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### convertImageObjects

The value of this option is a Boolean that if set to true will convert dr.image objects to a graphic file and the resulting file will be referenced by the locator child of the dr.image. If the option is set to false, dr.image objects will not be converted.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### convertPresentationObjects

The value of this option is a Boolean that if set to true will convert pr.slide objects to a graphic file and the resulting file will be referenced by the locator child of the pr.slide. If the option is set to false, pr.slide objects will not be converted.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### convertVectorObjects

The value of this option is a Boolean that if set to true will convert dr.drawing objects to a graphic file and the resulting file will be referenced by the locator child of the dr.drawing. If the option is set to false, dr.drawing objects will not be converted.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### delimiters

This option is applicable only to PDF files. Often, PDF 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 PDF filter indicates 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.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### flattenStyles

The value of this option is a Boolean that if set to true will flatten styles to eliminate the need to process the 'basedon' attribute. If the option is set to false, the 'basedon' attribute will be processed.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### noBitmapElements

The value of this option is a Boolean that if set to true will ensure that no children are produced for dr.image elements. If the option is set to false, dr.image elements will produce children.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### noChartElements

The value of this option is a Boolean that if set to true will ensure that no children are produced for ch.chart elements. If the option is set to false, ch.chart elements will produce children.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### noPresentationElements

The value of this option is a Boolean that if set to true will ensure that no children are produced for pr.slide elements. If the option is set to false, pr.slide elements will produce children.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### noVectorElements

The value of this option is a Boolean that if set to true will ensure that no children are produced for dr.drawing elements. If the option is set to false, dr.drawing elements will produce children.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### removeFontGroups

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.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

true

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### separateStyleTables

Enabling this option will cause the style_tables subtree to be streamed to a separate output unit. This flag is false by default.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

### subStreamRoots

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_5,style_tables
```

**Note:**

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

#### Export Product

XML Export

#### Data Type

xsd:string

#### 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

#### Equivalents

- C Client Implementation: TS_stringData
- JAVA Client Implementation: StringData

### useFullFilePaths

The value of this option is a Boolean that if set to true will ensure that the locators for externalized embeddings will contain full, absolute pathnames. If the option is set to false, full, absolute pathnames will not be included in the output.

#### Export Product

XML Export

#### Data Type

xsd:boolean

#### Default

false

#### Equivalents

- C Client Implementation: XSD_boolean
- JAVA Client Implementation: Boolean

## File System

This section pertains to file system options.

### fileAccess

This option supplies information to OIT when information is required to open an input file. This information may be the password of the file or a support file location.

Further information about how Transformation Server implements this option will be forthcoming.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export,

### readBufferSize

Used to define the number of bytes that 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.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The size of the buffer in kilobytes.

#### Default

2

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### memoryMappedInputSize

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 that the dwMMapBufferSize, the entire file will be read into memory, if not, it will be read in blocks defined by the dwReadBufferSize.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The size of the buffer in kilobytes.

#### Default

8192

#### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

### tempBufferSize

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.

#### Export Product

HTML Export, Image Export, PDF Export, Search Export, XML Export

#### Data Type

xsd:unsignedInt

#### Data

The size of the buffer in kilobytes.

#### Default

2048

### Equivalents

- C Client Implementation: XSD_unsignedInt
- JAVA Client Implementation: UnsignedInt

# B

# C/C++ Client Data Types

All options discussed in this chapter are described in detail in the Options documentation.

## Simple Types

- XSD_booleanL Binary data (true [non-zero] or false [0])
- XSD_byte: Short data between -128 and 127
- XSD_double: IEEE double-precision 64-bit floating point data
- XSD_float: IEEE single-precision 32-bit floating point data
- XSD_int: Long data between -2147483648 and 2147483647
- XSD_short: Integer data between -32768 and 32767
- XSD_signedInt: Integer data between -2147483648 and 2147483647
- XSD_string: A null-terminated character string
- XSD_unsignedByte: Unsigned, short data no greater than 255
- XSD_unsignedInt: Unsigned, long data no greater than 4294967295
- XSD_unsignedShort: Unsigned, short data no greater than 65535

## Complex Types

Complex types are listed by product:


- All Export Products
- HTML Export
- Search Export
- Image Export


### All Export Products

This section discusses information that is pertinent for all products.

#### TS_binaryData

Binary data. This data type takes the form of a structure, defined as follows:

```
typedef struct TS_binaryData
{
```

```
   XSD_unsignedByte * pData;
   XSD_unisginedInt   size;
} TS_binaryData;
```

- pData: Pointer to a buffer of bytes containing binary data
- size: The size of the data pointed to by pData

### TS_char*

This data type is a pointer to the standard C data type char*.

### TS_IOSpec

This data type is a structure that contains the full specification required for Transformation Server to open a particular data stream for input or output. In addition to a "specification," examples of which include a file system path or a URL, the structure also provides fields for the character set used in the specification and an identifier of the type of specification provided, for example, path or url. The TS_IOSpec structure is defined as follows:

```
typedef struct TS_IOSpec
{
   TS_stringData  spec;
   XSD_string     specType;
} TS_IOSpec;
```

- spec: The string containing the specification and the character set of that string
- specType: An identifier of the type of specification provided (path, url, or other type)

### TS_OutputList

This data type is a structure that is used by the TS_TransformResult data type. It contains a list of ID specs for Transformation Server output. The data type takes the form of a structure, defined as follows:

```
typedef struct TS_OutputList
{
   TS_IOSpec *     documents; /* array of output document
                                 specifications */
   XSD_unsignedInt size;      /* size of array */
} TS_OutputList;
```

### TS_stringArray

The TS_StringArray is used with the fileAccess option. It is an array of UTF-8 strings.

```
typedef struct TS_stringArray
{
   XSD_string **strings;     /* array of strings */
   XSD_unsignedInt size;
}  TS_stringArray;
```

### TS_stringData

This data type stores a text string along with an identifier of the character set used in the string.

Make sure that the charset field correctly identifies the character set used in the str field. For a list of available character sets, see TS_CharacterSetEnum. Note that unlike the SOAP API, the C API does not require or support strings that have been base64-encoded for transmission.

It takes the form of a data structure, defined as follows:

```
typedef struct TS_stringData
{
   TS_char*            str;
   TS_CharacterSetEnum charset;
} TS_stringData;
```

- str: A text string
- charset: An identifier of the character set used in the string stored in str

### TS_TransformResult

This data type is a structure that contains a human-readable result message and a list of the output documents created by the transformation. The structure is defined as follows:

```
typedef struct TS_TransformResult
{
   TSERR          resultCode;   /* numeric result code defined
                                   in TSERR.H */
   TS_stringData  resultString; /* result message;
                                   may be empty */
   TS_OutputList  outputList;   /* list of output documents */
} TS_TransformResult;
```

## HTML Export

The information in this section is pertinent for HTML Export only.

### OIT_AltLink

Valid for the altlink option. It takes the form of a data structure, defined as follows:

```
typedef struct OIT_AltLink
{
   XSD_string prev;
   XSD_string next;
} OIT_AltLink;
```

### OIT_DefaultFont

Valid for the SCCIDOPT_DEFAULTPRINTFONTdefaultFont option. The structure is defined as follows:

```
typedef struct OIT_DefaultFont
{
   TS_char*          fontName; /* UTF-8 string */
   XSD_unsignedShort height;
      // Currently the wAttr value of defaultprintfont is
      // ignored, so we don't need to either store it or set it.
} OIT_DefaultFont;
```

### OIT_FontFlags

Valid for the SCCOPT_EX_FONTFLAGSfontFlags option. It takes the form of a data structure, defined as follows:

```
typedef struct OIT_FontFlags
{
   XSD_boolean  suppressSize;
   XSD_boolean  suppressColor;
   XSD_boolean  suppressFace;
} OIT_FontFlags;
```

## Search Export

This information is valid for Search Export only.

### OIT_CharacterAttributes

This data type has been deprecated. The flags contained in it are now standalone XSD_boolean options.

```
typedef struct OIT_CharacterAttributes
{
   XSD_boolean bold;
   XSD_boolean italic;
   XSD_boolean underline;
   XSD_boolean doubleUnderline;
   XSD_boolean outline;
   XSD_boolean strikeout;
   XSD_boolean smallCaps;
   XSD_boolean allCaps;
   XSD_boolean hidden;
} OIT_CharacterAttributes;
```

### OIT_ParagraphAttributes

Valid for the SparagraphAttributes option. The structure is defined as follows:

```
typedef struct OIT_ParagraphAttributes
{
   XSD_boolean spacing;
   XSD_boolean height;
   XSD_boolean leftIndent;
   XSD_boolean rightIndent;
   XSD_boolean firstIndent;
} OIT_ParagraphAttributes;
```

### OIT_SearchMLFlags

This data type has been deprecated. The flags contained in it are now standalone XSD_boolean options.

```
typedef struct OIT_SearchMLFlags
{
   XSD_boolean   paragraphStyleNamesOn;
   XSD_boolean   embeddingsOn;
   XSD_boolean   xmlDeclarationOff;
   XSD_boolean   documentPropertiesOn;
   XSD_boolean   changeNumberToTextOn;
   XSD_boolean   suppressAttachments;
   XSD_boolean   suppressArchiveSubdocs;
} OIT_SearchMLFlags;
```

## Image Export

This information is valid for Image Export only.

### OIT_DefaultFont

Valid for the defaultFont option. The structure is defined as follows:

```
typedef struct OIT_DefaultFont
{
   TS_char*          fontName; /* UTF-8 string */
   XSD_unsignedShort height;
      // Currently the wAttr value of defaultprintfont is
      // ignored, so we don't need to either store it or set it.
} OIT_DefaultFont;
```

### OIT_DefaultMargins

Valid for the defaultMargins option. It takes the form of a data structure, defined as follows:

```
typedef struct OIT_DefaultMargins
{
   XSD_unsignedInt top;
   XSD_unsignedInt bottom;
   XSD_unsignedInt left;
   XSD_unsignedInt right;
} OIT_DefaultMargins;
```

### OIT_TiffOptions

Valid for the tiffOptions option. The structure is defined as follows:

```
typedef struct OIT_TiffOptions
{
   OIT_TiffColorSpaceEnum  colorSpace;
   OIT_TiffCompressionEnum compression;
   OIT_TiffByteOrderEnum   byteOrder;
   OIT_TiffFillOrderEnum   fillOrder;
   XSD_boolean             createOneFile;
} OIT_TiffOptions;
```

# Enumerations

Complex types are listed by product:

- All Export Products
- HTML Export
- Search Export
- Image Export
- PDF Export
- XML Export

## All Export Products

This information is valid for all products.

### OIT_DefaultInputCharSetEnum

Valid for the defaultInputCharset option. The enumeration is defined as follows:

```
typedef enum OIT_DefaultInputCharSetEnum
{
   oit_jis = 145,
   oit_euc_jp,
   oit_cns11643_1,
   oit_euc_cns_1,
   oit_cns11643_2,
   oit_euc_cns_2,
   oit_ksc1987,
   oit_gb2312,
   oit_ebcdic37,
   oit_ebcdic273,
   oit_ebcdic274,
   oit_ebcdic277,
```

```
   oit_ebcdic278,
   oit_ebcdic280,
   oit_ebcdic282,
   oit_ebcdic284,
   oit_ebcdic285,
   oit_ebcdic297,
   oit_ebcdic500,
   oit_ebcdic1026,
   oit_ansi437,
   oit_ansi737,
   oit_ansi850,
   oit_ansi852,
   oit_ansi855,
   oit_ansi857,
   oit_ansi860,
   oit_ansi861,
   oit_ansi863,
   oit_ansi865,
   oit_ansi866,
   oit_ansi869,
   oit_ansi874,
   oit_ansi932,
   oit_ansi936,
   oit_ansi949,
   oit_ansi950,
   oit_ansi1250,
   oit_ansi1251,
   oit_ansi1252,
   oit_ansi1253,
   oit_ansi1254,
   oit_ansi1255,
   oit_ansi1256,
   oit_ansi1257,
   oit_iso8859_1,
   oit_iso8859_2,
   oit_iso8859_3,
   oit_iso8859_4,
   oit_iso8859_5,
   oit_iso8859_6,
   oit_iso8859_7,
   oit_iso8859_8,
   oit_iso8859_9,
   oit_macroman,
   oit_macromancroatian,
   oit_macromanromanian,
   oit_macromanturkish,
   oit_macromanicelandic,
   oit_maccyrillic,
   oit_macgreek,
   oit_maclatin2,
   oit_hebrew,
   oit_arabic,
   oit_macjis,
   oit_hproman8,
   oit_bidi_oldcode,
   oit_bidi_pc8,
   oit_bidi_e0,
   oit_htmlkoi8,
   oit_jis_roman,
} OIT_DefaultInputCharSetEnum;
```

### OIT_FallbackFormatEnum

Valid for the fallbackFormat option. The enumeration is defined as follows:

```
typedef enum OIT_FallbackFormatEnum
{
   oit_ANSI_7,
   oit_ANSI_8,
```

```
   oit_ASCII_7,
   oit_ASCII_8,
   oit_Big5,
   oit_EUC_KR,
   oit_EUC_JP,
   oit_GB2312,
   oit_hebrew_old_code,
   oit_ISO_2022_JP,
   oit_ISO_8859_2,
   oit_ISO_8859_6,
   oit_ISO_10646_UCS_2,
   oit_KOI8_R,
   oit_Shift_JIS,
   oit_UTF_7
   oit_UTF_8
   oit_windows_874,
   oit_windows_1250,
   oit_windows_1251,
   oit_windows_1252,
   oit_windows_1253,
   oit_windows_1254,
   oit_windows_1255,
   oit_windows_1256,
   oit_windows_1257,
   oit_x_Mac_roman_7,
   oit_x_Mac_roman,
   oit_noFallbackFormat
} OIT_FallbackFormatEnum;
```

### OIT_DocumentMemoryModeEnum

Valid for the documentMemoryMode option. The enumeration is defined as follows:

```
typedef enum OIT_DocumentMemoryModeEnum
{
   oit_smallestMode,
   oit_smallMode,
   oit_mediumMode,
   oit_largeMode,
   oit_largestMode,
} OIT_DocumentMemoryModeEnum;
```

## HTML Export

This information is valid for HTML Export only.

### OIT_CharacterByteOrderEnum

Valid for the characterByteOrder option. The enumeration is defined as follows:

```
typedef enum OIT_CharacterByteOrderEnum
{
   oit_big_endian,
   oit_little_endian,
   oit_template_order
} OIT_CharacterByteOrderEnum;
```

### OIT_ComplianceEnum

Valid for the compliance option. The enumeration is defined as follows:

```
typedef enum OIT_ComplianceEnum
{
   oit_none,
   oit_well_formed,
   oit_strictDTD
} OIT_ComplianceEnum;
```

### OIT_EmailHeaderOutputEnum

Valid for the emailHeaderOutput option. The enumeration is defined as follows:

```
typedef enum OIT_EmailHeaderOutputEnum
{
   oit_emailHeaderStandard,
   oit_emailHeaderAll
   oit_emailHeaderNone
} OIT_EmailHeaderOutputEnum;
```

### OIT_ExtractEmbeddedFilesEnum

Valid for the extractEmbeddedFiles option. The enumeration is defined as follows:

```
typedef enum OIT_ExtractEmbeddedFilesEnum
{
   oit_ignoreFiles,
   oit_convertFiles,
   oit_extractFiles,
} OIT_ExtractEmbeddedFilesEnum;
```

### OIT_FlavorEnum

Valid for the flavor option. The enumeration is defined as follows:

```
typedef enum OIT_FlavorEnum
{
   oit_generic_html,
   oit_generic_wireless_html,
   oit_html2_0,
   oit_html3_0,
   oit_html4_0,
   oit_netscape3_0,
   oit_netscape4_0,
   oit_internetExplorer3_0,
   oit_internetExplorer4_0,
   oit_avantGo3_3_palm,
   oit_avantGo3_3_palm_noTables,
   oit_avantGo3_3_winCE,
   oit_avantGo3_3_winCE_noTables,
   oit_webClipping1_1,
   oit_webClipping1_1_noTables,
   oit_chtml2_0,
   oit_hdml3_0,
   oit_text,
   oit_wml1_1,
   oit_wml1_1_withTables,
   oit_wml2_0,
   oit_xhtml_basic1_0,
   oit_xhtml_basic1_0_noTables
} OIT_FlavorEnum;
```

### OIT_GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicSizeMethodEnum
{
   oit_smooth,
   oit_quick,
   oit_smoothGray
} OIT_GraphicSizeMethodEnum;
```

### OIT_GraphicTypeEnum

Valid for the graphicType option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicTypeEnum
{
   oit_bmp,
   oit_gif,
   oit_jpeg,
   oit_noGraphics,
   oit_png,
   oit_wbmp
} OIT_GraphicTypeEnum;
```

### OIT_GridAdvanceEnum

Valid for the gridAdvance option. The enumeration is defined as follows:

```
typedef enum OIT_GridAdvanceEnum
{
   oit_advanceAcross,
   oit_advanceDown
} OIT_GridAdvanceEnum;
```

### OIT_ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
typedef enum OIT_ReorderMethodEnum
{
   oit_reorderOff = oit_emailHeaderAll +1,
   oit_reorderLeftToRight,
   oit_reorderRightToLeft
} OIT_ReorderMethodEnum;
```

### OIT_SpreadSheetBordersEnum

Valid for the spreadsheetBorders option. The enumeration is defined as follows:

```
typedef enum OIT_SpreadsheetBordersEnum
{
   oit_createBorderIfMissing,
   oit_bordersOff,
   oit_useSourceBorders
} OIT_SpreadsheetBordersEnum;
```

### TS_CharacterSetEnum

Valid for the outputCharacterSet option. The enumeration is defined as follows:

```
typedef enum TS_CharacterSetEnum
{
   ts_ISO_8859_1        = 0x00080101,
   ts_ISO_8859_2        = 0x00080102,
   ts_ISO_8859_3        = 0x00080103,
   ts_ISO_8859_4        = 0x00080104,
   ts_ISO_8859_5        = 0x00080105,
   ts_ISO_8859_6        = 0x00080106,
   ts_ISO_8859_7        = 0x00080107,
   ts_ISO_8859_8        = 0x00080108,
   ts_ISO_8859_9        = 0x00080109,
   ts_x_Mac_roman       = 0x80000100,
   ts_x_Mac_ce          = 0x80070100,
   ts_x_Mac_Greek       = 0x80060100,
   ts_x_Mac_Cyrillic    = 0x80050100,
   ts_x_Mac_Turkish     = 0x80030100,
   ts_GB2312            = 0x0f050000,
   ts_Big5              = 0x13b60000,
   ts_Shift_JIS         = 0x13a40000,
   ts_KOI8_R            = 0x000a0101,
   ts_windows_1250      = 0x14e20100,
   ts_windows_1251      = 0x14e30100,
   ts_windows_1252      = 0x14e40100,
```

```
   ts_windows_1253      = 0x14e50100,
   ts_windows_1254      = 0x14e60100,
   ts_windows_1255      = 0x14e70100,
   ts_windows_1256      = 0x14e80100,
   ts_windows_1257      = 0x14e90100,
   ts_EUC_KR            = 0x13b50000,
   ts_EUC_JP            = 0x0f0d0000,
   ts_ISO_2022_JP       = 0x0f0c0000,
   ts_windows_874       = 0x136a0100,
   ts_UTF_7             = 0x000b000b,
   ts_UTF_8             = 0x000b000b,
   ts_ISO_10646_UCS_2   = 0x14b00000,
   ts_x_Charset_Unknown = 0
};
```

## Search Export

This information is valid for Search Export only.

### OIT_OleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

```
typedef enum OIT_OleEmbeddingsEnum
{
   oit_processStandard, /* Process embeddings that are known standard
embeddings*/
   oit_processAll,      /* Process all embeddings in the file */
   oit_processNone      /* Process none of the embeddings in the file */
} OIT_ProcessOleEmbeddingsEnum;
```

### OIT_SearchMLUnmappedTextEnum

Valid for the unmappedText option. The enumeration is defined as follows:

```
typedef enum OIT_SearchMLUnmappedTextEnum
(
   oit_justUnmappedText,
   oit_noUnmappedText,
   oit_bothUnmappedText
} OIT_SearchMLUnmappedTextEnum
```

### OIT_XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option. The enumeration is defined as follows:

```
typedef enum OIT_XmlDefinitionMethodEnum
{
   oit_dtd,
   oit_noDefinition,
   oit_xsd
} OIT_XmlDefinitionMethodEnum;
```

## Image Export

This information is valid for Image Export only.

### OIT_DatabaseFitToPageEnum

Valid for the databaseFitToPage option. The enumeration is defined as follows:

```
typedef enum OIT_DatabaseFitToPageEnum
{
   oit_dbNoScaling,
   oit_dbFitToPage,
   oit_dbFitToWidth,
```

```
   oit_dbFitToHeight,
} OIT_DatabaseFitToPageEnum;
```

### OIT_EmailHeaderOutputEnum

Valid for the emailHeaderOutput option. The enumeration is defined as follows:

```
typedef enum OIT_EmailHeaderOutputEnum
{
   oit_emailHeaderStandard,
   oit_emailHeaderAll
   oit_emailHeaderNone
} OIT_EmailHeaderOutputEnum;
```

### OIT_GraphicCroppingEnum

Valid for the graphicCropping option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicCroppingEnum
{
   oit_noCropping,
   oit_cropToContent
} OIT_GraphicCroppingEnum;
```

### OIT_GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicSizeMethodEnum
{
   oit_smooth,
   oit_quick,
   oit_smoothGray
} OIT_GraphicSizeMethodEnum;
```

### OIT_GraphicWatermarkScaleTypeEnum

Valid for the graphicWatermarkScaleType option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicWatermarkScaleTypeEnum
{
  oit_scaleWatermarkOff,
  oit_scaleWatermarkByPercent
} OIT_GraphicWatermarkScaleTypeEnum;
```

### OIT_MimeHeaderOutputEnum

Valid for the mimeHeaderOutput option. The enumeration is defined as follows:

```
typedef enum OIT_MimeHeaderOutputEnum
{
   oit_all,
   oit_standard,
} OIT_MimeHeaderOutputEnum
```

### OIT_ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
typedef enum OIT_ReorderMethodEnum
{
   oit_reorderOff = oit_emailHeaderAll +1,
   oit_reorderLeftToRight,
   oit_reorderRightToLeft
} OIT_ReorderMethodEnum;
```

### OIT_SpreadsheetFitToPageEnum

Valid for the EspreadsheetFitToPage option. The enumeration is defined as follows:

```
typedef enum OIT_SpreadsheetFitToPageEnum
{
   oit_ssNoScaling,
   oit_ssFitToPage,
   oit_ssFitToWidth,
   oit_ssFitToHeight,
   oit_ssScaleByPercentage,
   oit_ssFitToPages
} OIT_SpreadsheetFitToPageEnum;
```

### OIT_SpreadsheetPageDirectionEnum

Valid for the spreadsheetPageDirection option. The enumeration is defined as follows:

```
typedef enum OIT_SpreadsheetPageDirectionEnum
{
   oit_downThenAcross,
   oit_acrossThenDown
} OIT_SpreadsheetPageDirectionEnum;
```

### OIT_TiffByteOrderEnum

Part of the OIT_TiffOptions structure. The enumeration is defined as follows:

```
typedef enum OIT_TiffByteOrderEnum
{
   oit_tiff_little_endian,
   oit_tiff_big_endian
} OIT_TiffByteOrderEnum;
```

### OIT_TiffColorSpaceEnum

Part of the OIT_TiffOptions structure. The enumeration is defined as follows:

```
typedef enum OIT_TiffColorSpaceEnum
{
   oit_1Bit,
   oit_8Bit,
   oit_24Bit
} OIT_TiffColorSpaceEnum;
```

### OIT_TiffCompressionEnum

Part of the OIT_TiffOptions structure. The enumeration is defined as follows:

```
typedef enum OIT_TiffCompressionEnum
{
   oit_noCompression,
   oit_packbits,
   oit_LZW,
   oit_CCITT_1D,
   oit_CCITT_Group3_Fax,
   oit_CCITT_Group4_Fax
} OIT_TiffCompressionEnum;
```

### OIT_TiffFillOrderEnum

Part of the OIT_TiffOptions structure. The enumeration is defined as follows:

```
typedef enum OIT_TiffFillOrderEnum
{
   oit_fillOrder_1,
   oit_fillOrder_2
} OIT_TiffFillOrderEnum;
```

## PDF Export

This information is valid for PDF Export only.

### OIT_DefaultPageUnitsEnum

Valid for the defaultPageUnits option. The enumeration is defined as follows:

```
typedef enum OIT_DefaultPageUnitsEnum
{
  oit_inches,
  oit_points,
  oit_centimeters,
  oit_picas,
} OIT_DefaultPageUnitsEnum;
```

### OIT_EmailHeaderOutputEnum

Valid for the emailHeaderOutput option. The enumeration is defined as follows:

```
typedef enum OIT_EmailHeaderOutputEnum
{
   oit_emailHeaderStandard,
   oit_emailHeaderAll
   oit_emailHeaderNone
} OIT_EmailHeaderOutputEnum;
```

### OIT_ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
typedef enum OIT_ReorderMethodEnum
{
   oit_reorderOff = oit_emailHeaderAll +1,
   oit_reorderLeftToRight,
   oit_reorderRightToLeft
} OIT_ReorderMethodEnum;
```

### OIT_WatermarkPositionEnum

Valid for the watermarkPosition option. The enumeration is defined as follows:

```
typedef enum OIT_WatermarkPostionEnum
{
   oit_centerOfPage
} OIT_WatermarkPostionEnum;
```

### OIT_WatermarkScalingEnum

Valid for the watermarkScaling option. The enumeration is defined as follows:

```
typedef enum OIT_WatermarkScalingEnum
{
   oit_pdfNoMap,
   oit_pdfFitToPage,
   oit_pdfScale,
} OIT_WatermarkScalingEnum;
```

## XML Export

This information is valid for XML Export only.

### OIT_GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicSizeMethodEnum
{
   oit_smooth,
   oit_quick,
   oit_smoothGray
} OIT_GraphicSizeMethodEnum;
```

### OIT_GraphicTypeEnum

Valid for the graphicType option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicTypeEnum
{
   oit_bmp,
   oit_gif,
   oit_jpeg,
   oit_noGraphics,
   oit_png,
   oit_wbmp
} OIT_GraphicTypeEnum;
```

### OIT_OleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

```
typedef enum OIT_OleEmbeddingsEnum
{
   oit_processStandard, /* Process embeddings that are known standard
embeddings*/
   oit_processAll,      /* Process all embeddings in the file */
   oit_processNone      /* Process none of the embeddings in the file */
} OIT_ProcessOleEmbeddingsEnum;
```

### OIT_ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
typedef enum OIT_ReorderMethodEnum
{
   oit_reorderOff = oit_emailHeaderAll +1,
   oit_reorderLeftToRight,
   oit_reorderRightToLeft
} OIT_ReorderMethodEnum;
```

### OIT_XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option. The enumeration is defined as follows:

```
typedef enum OIT_XmlDefinitionMethodEnum
{
   oit_dtd,
   oit_noDefinition,
   oit_xsd
} OIT_XmlDefinitionMethodEnum;
```

# C

# Java Client Data Types

All options discussed in this chapter are described in detail in the Options documentation.

## Simple Types

- Boolean: Binary data (true [non-zero] or false [0])
- Byte: Short data between -128 and 127
- Double: IEEE double-precision 64-bit floating point data
- Float: IEEE single-precision 32-bit floating point data
- hexBinary: Arbitrary hex-encoded binary data
- Integer: Long data between -2147483648 and 2147483647
- Short: Integer data between -32768 and 32767
- SignedInt: Integer data between -2147483648 and 2147483647
- String: A null-terminated character string
- UnsignedByte: Unsigned, short data no greater than 255
- UnsignedInt: Unsigned, long data no greater than 4294967295
- UnsignedShort: Unsigned, short data no greater than 65535

## Complex Types

This topic is listed by product:



### All Products

This information pertains to all products.

### IOSpec

This data type is a class that contains the full specification required for Transformation Server to open a particular data stream for input or output. In addition to a "specification," examples of which include a file system path or a URL, the class also provides fields for the character set used in the specification and an identifier of the type of specification provided (for example, path or url). The class is defined as follows:

```
public IOSpec(StringData spec, String specType) {
   this.spec = spec;
   this.specType = specType;
}

public StringData getSpec() {
   return spec;
}

public java.lang.String getSpecType() {
   return specType;
}
```

- spec: The string containing the specification and the character set of that string
- specType: An identifier of the type of specification provided (path, url, or other type)

### StringData

This data type is a class that stores a text string along with an identifier of the character set used in the string.

---

**Note:**

Make sure the charset field correctly identifies the character set used in the str field. For a list of available character sets, see CharacterSetEnum. Note that unlike the SOAP API, the Java API does not require or support strings that have been base64-encoded for transmission.

---

The class is defined as follows:

```
public StringData(String str, CharacterSetEnum charSet) {
   this.str = str;
   this.charset = charSet;
}

public String getStr() {
   return str;
}

public CharacterSetEnum getCharset() {
   return charset;
}
```

- str: A text string
- charset: An identifier of the character set used in the string stored in str

### TransformReponse

This data type is a class that contains a human-readable result message and a list of the output documents created by the transformation.

The class is defined as follows:

```
public TransformResponse(String str, CharacterSetEnum charSet) {
   this.str = str;
   this.charset = charSet;
}

public long getResult() {
   return str;
}
public IOSpec getResultDocs() {
   return str;
}
public StringData getResultMsg() {
   return str;
}
public void setResult() {
   return str;
}
public void setResultDocs() {
   return str;
}
public void setResultMsg() {
   return str;
}
```

## HTML Export

The following information pertains to HTML Export.

### AltLink

Valid for the altlink option. The class is defined as follows:

```
public AltLink() {
}

public String getPrev() {
   return prev;
}

public void setPrev(String prev) {
   this.prev = prev;
}

public String getNext() {
   return next;
}

public void setNext(String next) {
   this.next = next;
}
```

### DefaultFont

Valid for the defaultFont option. The class is defined as follows:

```
public DefaultFont() {
}

public java.lang.String getFontName() {
   return fontname;
```

```
}

public int getHeight() {
   return height;
}

public void setFontName(java.lang.String fontName) {
   if (fontName == null) {
      this.fontName = "";
   } else {
      this.fontName = fontName;
   }
}

public void setHeight(int height) {
   this.height = height;
};
```

### FontFlags

Valid for the fontFlags option. The class is defined as follows:

```
public FontFlags(Boolean size, Boolean color, Boolean face) {
   this.suppressSize = size;
   this.suppressColor = color;
   this.suppressFace = face;
}

public java.lang.Boolean getSuppressSize() {
   return suppressSize;
}

public java.lang.Boolean getSuppressColor() {
   return suppressColor;
}

public java.lang.Boolean getSuppressFace() {
   return suppressFace;
}
```

## Search Export

The following information applies to Search Export.

### CharacterAttributes

This data type has been deprecated. The flags contained in it are now standalone Boolean options.

```
public CharacterAttributes() {
}

public Boolean getBold() {
   return bold;
}

public void setBold(Boolean bold) {
   this.bold = bold;
}

public Boolean getItalic() {
   return italic;
}

public void setItalic(Boolean italic) {
   this.italic = italic;
}
```

```
public Boolean getUnderline() {
   return underline;
}

public void setUnderline(Boolean underline) {
   this.underline = underline;
}

public Boolean getDoubleUnderline() {
   return doubleUnderline;
}

public void setDoubleUnderline(Boolean doubleUnderline) {
   this.doubleUnderline = doubleUnderline;
}

public Boolean getOutline() {
   return outline;
}

public void setOutline(Boolean outline) {
   this.outline = outline;
}

public Boolean getStrikeout() {
   return strikeout;
}

public void setStrikeout(Boolean strikeout) {
   this.strikeout = strikeout;
}

public Boolean getSmallCaps() {
   return smallCaps;
}

public void setSmallCaps(Boolean smallCaps) {
   this.smallCaps = smallCaps;
}

public Boolean getAllCaps() {
   return allCaps;
}

public void setAllCaps(Boolean allCaps) {
   this.allCaps = allCaps;
}

public Boolean getHidden() {
   return hidden;
}

public void setHidden(Boolean hidden) {
   this.hidden = hidden;
}
```

### ParagraphAttributes

Valid for the paragraphAttributes option. The class is defined as follows:

```
public ParagraphAttributes() {
}

public Boolean getSpacing() {
   return spacing;
}

public void setSpacing(Boolean spacing) {
   this.spacing = spacing;
```

```
}

public Boolean getHeight() {
   return height;
}

public void setHeight(Boolean height) {
   this.height = height;
}

public Boolean getLeftIndent() {
   return leftIndent;
}

public void setLeftIndent(Boolean leftIndent) {
   this.leftIndent = leftIndent;
}

public Boolean getRightIndent() {
   return rightIndent;
}

public void setRightIndent(Boolean rightIndent) {
   this.rightIndent = rightIndent;
}

public Boolean getFirstIndent() {
   return firstIndent;
}

public void setFirstIndent(Boolean firstIndent) {
   this.firstIndent = firstIndent;
}
```

### SearchMLFlags

This data type has been deprecated. The flags contained in it are now standalone Boolean options.

## Image Export

This information applies to Image Export.

### DefaultFont

Valid for the defaultFont option. The class is defined as follows:

```
public DefaultFont() {
}

public java.lang.String getFontName() {
   return fontname;
}

public int getHeight() {
   return height;
}

public void setFontName(java.lang.String fontName) {
   if (fontName == null) {
      this.fontName = "";
   } else {
      this.fontName = fontName;
   }
}

public void setHeight(int height) {
```

```
   this.height = height;
};
```

### DefaultMargins

Valid for the defaultMargins option. The class is defined as follows:

```
public DefaultMargins() {
}

public Long getTop() {
   return top;
}

public void setTop(Long top) {
   this.top = top;
}

public Long getBottom() {
   return bottom;
}

public void setBottom(Long bottom) {
   this.bottom = bottom;
}

public Long getLeft() {
   return left;
}

public void setLeft(Long left) {
   this.left = left;
}

public Long getRight() {
   return right;
}

public void setRight(Long right) {
   this.right = right;
}
```

### TiffOptions

Valid for the tiffOptions option. The class is defined as follows:

```
public TiffOptions() {
}

public TiffColorSpaceEnum getColorSpace() {
   return colorSpace;
}

public void setColorSpace(TiffColorSpaceEnum colorSpace) {
   this.colorSpace = colorSpace;
}

public TiffCompressionEnum getCompression() {
   return compression;
}

public void setCompression(TiffCompressionEnum compression) {
   this.compression = compression;
}

public TiffByteOrderEnum getByteOrder() {
   return byteOrder;
}
```

```
public void setByteOrder(TiffByteOrderEnum byteOrder) {
   this.bytorder = byteOrder;
}

public TiffFillOrderEnum getFillOrder() {
   return (fillOrder);
}

public void setFillOrder(TiffFillOrderEnum fillOrder) {
   this.fillOrder = fillOrder;
}

public Boolean getCreateOneFile() {
   return createOneFile;
}

public void setCreateOneFile(Boolean createOneFile) {
   this.createOneFile = createOneFile;
}
```

# Enumerations

This topic has these sections:



## All Export

This information applies to all products.

### DefaultInputCharSetEnum

Valid for the defaultInputCharset option. The class DefaultInputCharsetEnum defines the following static members:

```
public static final DefaultInputCharSetEnum JIS = new Default
InputCharSetEnum("JIS");
public static final DefaultInputCharSetEnum EUC_JP = new Default
InputCharSetEnum("EUC-JP");
public static final DefaultInputCharSetEnum CNS11643_1 = new Default
InputCharSetEnum("CNS11642-1");
public static final DefaultInputCharSetEnum EUC_CNS_1 = new Default
InputCharSetEnum("EUC-CNS-1");
public static final DefaultInputCharSetEnum CNS11643_2 = new Default
InputCharSetEnum("CNS11643-2");
public static final DefaultInputCharSetEnum EUC_CNS_2 = new Default
InputCharSetEnum("EUC-CNS-2");
public static final DefaultInputCharSetEnum KSC1987 = new Default
InputCharSetEnum("KSC1987");
public static final DefaultInputCharSetEnum GB2312 = new Default
InputCharSetEnum("GB2312");
public static final DefaultInputCharSetEnum JIS1978 = new Default
InputCharSetEnum("JIS1978");
public static final DefaultInputCharSetEnum JIS1983 = new Default
InputCharSetEnum("JIS1983");
```

```
public static final DefaultInputCharSetEnum JIS1990 = new Default
InputCharSetEnum("JIS1990");
public static final DefaultInputCharSetEnum EBCDIC37 = new Default
InputCharSetEnum("EBCDIC37");
public static final DefaultInputCharSetEnum EBCDIC273 = new Default
InputCharSetEnum("EBCDIC273");
public static final DefaultInputCharSetEnum EBCDIC274 = new Default
InputCharSetEnum("EBCDIC274");
public static final DefaultInputCharSetEnum EBCDIC277 = new Default
InputCharSetEnum("EBCDIC277");
public static final DefaultInputCharSetEnum EBCDIC278 = new Default
InputCharSetEnum("EBCDIC278");
public static final DefaultInputCharSetEnum EBCDIC280 = new Default
InputCharSetEnum("EBCDIC280");
public static final DefaultInputCharSetEnum EBCDIC282 = new Default
InputCharSetEnum("EBCDIC282");
public static final DefaultInputCharSetEnum EBCDIC284 = new Default
InputCharSetEnum("EBCDIC284");
public static final DefaultInputCharSetEnum EBCDIC285 = new Default
InputCharSetEnum("EBCDIC285");
public static final DefaultInputCharSetEnum EBCDIC297 = new Default
InputCharSetEnum("EBCDIC297");
public static final DefaultInputCharSetEnum EBCDIC500 = new Default
InputCharSetEnum("EBCDIC500");
public static final DefaultInputCharSetEnum EBCDIC1026 = new Default
InputCharSetEnum("EBCDIC1026");
public static final DefaultInputCharSetEnum DCA = new Default
InputCharSetEnum("DCA");
public static final DefaultInputCharSetEnum ANSI0 = new Default
InputCharSetEnum("ANSI0");
public static final DefaultInputCharSetEnum ASCII = new Default
InputCharSetEnum("ASCII");
public static final DefaultInputCharSetEnum ANSI437 = new Default
InputCharSetEnum("ANSI437");
public static final DefaultInputCharSetEnum ANSI737 = new Default
InputCharSetEnum("ANSI737");
public static final DefaultInputCharSetEnum ANSI850 = new Default
InputCharSetEnum("ANSI850");
public static final DefaultInputCharSetEnum ANSI852 = new Default
InputCharSetEnum("ANSI852");
public static final DefaultInputCharSetEnum ANSI855 = new Default
InputCharSetEnum("ANSI855");
public static final DefaultInputCharSetEnum ANSI857 = new Default
InputCharSetEnum("ANSI857");
public static final DefaultInputCharSetEnum ANSI860 = new Default
InputCharSetEnum("ANSI860");
public static final DefaultInputCharSetEnum ANSI861 = new Default
InputCharSetEnum("ANSI861");
public static final DefaultInputCharSetEnum ANSI863 = new Default
InputCharSetEnum("ANSI863");
public static final DefaultInputCharSetEnum ANSI865 = new Default
InputCharSetEnum("ANSI865");
public static final DefaultInputCharSetEnum ANSI866 = new Default
InputCharSetEnum("ANSI866");
public static final DefaultInputCharSetEnum ANSI869 = new Default
InputCharSetEnum("ANSI869");
public static final DefaultInputCharSetEnum ANSI874 = new Default
InputCharSetEnum("ANSI874");
public static final DefaultInputCharSetEnum ANSI932 = new Default
InputCharSetEnum("ANSI932");
public static final DefaultInputCharSetEnum ANSI936 = new Default
InputCharSetEnum("ANSI936");
public static final DefaultInputCharSetEnum ANSI949 = new Default
InputCharSetEnum("ANSI949");
public static final DefaultInputCharSetEnum ANSI950 = new Default
InputCharSetEnum("ANSI950");
public static final DefaultInputCharSetEnum THAINOVELL = new Default
InputCharSetEnum("THAINOVELL");
public static final DefaultInputCharSetEnum ANSI1250 = new Default
```

```
InputCharSetEnum("ANSI1250");
public static final DefaultInputCharSetEnum ANSI1251 = new Default
InputCharSetEnum("ANSI1251");
public static final DefaultInputCharSetEnum ANSI1252 = new Default
InputCharSetEnum("ANSI1252");
public static final DefaultInputCharSetEnum ANSI1253 = new Default
InputCharSetEnum("ANSI1253");
public static final DefaultInputCharSetEnum ANSI1254 = new Default
InputCharSetEnum("ANSI1254");
public static final DefaultInputCharSetEnum ANSI1255 = new Default
InputCharSetEnum("ANSI1255");
public static final DefaultInputCharSetEnum ANSI1256 = new Default
InputCharSetEnum("ANSI1256");
public static final DefaultInputCharSetEnum ANSI1257 = new Default
InputCharSetEnum("ANSI1257");
public static final DefaultInputCharSetEnum HWP_HANGUL = new Default
InputCharSetEnum("HWP_HANGUL");
public static final DefaultInputCharSetEnum UNICODE = new Default
InputCharSetEnum("UNICODE");
public static final DefaultInputCharSetEnum PDFCID_JAPAN1_H = new Default
InputCharSetEnum("PDFCID-JAPAN1-H");
public static final DefaultInputCharSetEnum PDFCID_JAPAN1_V = new Default
InputCharSetEnum("PDFCID-JAPAN1-V");
public static final DefaultInputCharSetEnum PDFCID_JAPAN2 = new Default
InputCharSetEnum("PDFCID-JAPAN2");
public static final DefaultInputCharSetEnum PDFCID_GB1 = new Default
InputCharSetEnum("PDFCID-GB1");
public static final DefaultInputCharSetEnum PDFCID_CNS_H = new Default
InputCharSetEnum("PDFCID-CNS-H");
public static final DefaultInputCharSetEnum PDFCID_CNS_V = new Default
InputCharSetEnum("PDFCID-CNS-V");
public static final DefaultInputCharSetEnum PDFCID_KOREA1 = new Default
InputCharSetEnum("PDFCID_KOREA1");
public static final DefaultInputCharSetEnum ISO8859_1 = new Default
InputCharSetEnum("ISO8859_1");
public static final DefaultInputCharSetEnum ISO8859_2 = new Default
InputCharSetEnum("ISO8859_2");
public static final DefaultInputCharSetEnum ISO8859_3 = new Default
InputCharSetEnum("ISO8859_3");
public static final DefaultInputCharSetEnum ISO8859_4 = new Default
InputCharSetEnum("ISO8859_4");
public static final DefaultInputCharSetEnum ISO8859_5 = new Default
InputCharSetEnum("ISO8859_5");
public static final DefaultInputCharSetEnum ISO8859_6 = new Default
InputCharSetEnum("ISO8859_6");
public static final DefaultInputCharSetEnum ISO8859_7 = new Default
InputCharSetEnum("ISO8859_7");
public static final DefaultInputCharSetEnum ISO8859_8 = new Default
InputCharSetEnum("ISO8859_8");
public static final DefaultInputCharSetEnum ISO8859_9 = new Default
InputCharSetEnum("ISO8859_9");
public static final DefaultInputCharSetEnum MACROMAN = new Default
InputCharSetEnum("MACROMAN");
public static final DefaultInputCharSetEnum MACROMANCROTIA = new Default
InputCharSetEnum("MACROMANCROTIA");
public static final DefaultInputCharSetEnum macromanromanian = new Default
InputCharSetEnum("MACROMANROMANIA");
public static final DefaultInputCharSetEnum macromanturkish = new Default
InputCharSetEnum("MACROMANTURKISH");
public static final DefaultInputCharSetEnum macromanicelandic = new Default
InputCharSetEnum("MACROMANICELANDIC");
public static final DefaultInputCharSetEnum maccyrillic = new Default
InputCharSetEnum("MACCYRILLIC");
public static final DefaultInputCharSetEnum macgreek = new Default
InputCharSetEnum("MACGREEK");
public static final DefaultInputCharSetEnum maclatin2 = new Default
InputCharSetEnum("MACLATIN2");
public static final DefaultInputCharSetEnum greek2 = new Default
InputCharSetEnum("GREEK2");
```

```
public static final DefaultInputCharSetEnum hebrew = new Default
InputCharSetEnum("HEBREW");
public static final DefaultInputCharSetEnum arabic = new Default
InputCharSetEnum("ARABIC");
public static final DefaultInputCharSetEnum macjis = new Default
InputCharSetEnum("MACJIS");
public static final DefaultInputCharSetEnum winsymbol = new Default
InputCharSetEnum("WINSYMBOL");
public static final DefaultInputCharSetEnum macsymbol = new Default
InputCharSetEnum("MACSYMBOL");
public static final DefaultInputCharSetEnum placeholder = new Default
InputCharSetEnum("PLACEHOLDER");
public static final DefaultInputCharSetEnum mslinedraw = new Default
InputCharSetEnum("MSLINEDRAW");
public static final DefaultInputCharSetEnum zapfdingbats = new Default
InputCharSetEnum("ZAPFDINGBATS");
public static final DefaultInputCharSetEnum wparabic = new Default
InputCharSetEnum("WPARABIC");
public static final DefaultInputCharSetEnum wparabicscript = new Default
InputCharSetEnum("WPARABICSCRIPT");
public static final DefaultInputCharSetEnum wpboxdrawing = new Default
InputCharSetEnum("WPBOXDRAWING");
public static final DefaultInputCharSetEnum wpcyrillica = new Default
InputCharSetEnum("WPCYRILLICA");
public static final DefaultInputCharSetEnum wpcyrillicb = new Default
InputCharSetEnum("WPCYRILLICB");
public static final DefaultInputCharSetEnum wpgreek = new Default
InputCharSetEnum("WPGREEK");
public static final DefaultInputCharSetEnum wphebrewdavid = new Default
InputCharSetEnum("WPHEBREWDAVID");
public static final DefaultInputCharSetEnum wpiconicsymbolsa = new Default
InputCharSetEnum("WPICONICSYMBOLSA");
public static final DefaultInputCharSetEnum wpiconicsymbolsb = new Default
InputCharSetEnum("WPICONICSYMBOLSB");
public static final DefaultInputCharSetEnum wpjapanese = new Default
InputCharSetEnum("WPJAPANESE");
public static final DefaultInputCharSetEnum wpmatha = new Default
InputCharSetEnum("WPMATHA");
public static final DefaultInputCharSetEnum wpmathb = new Default
InputCharSetEnum("WPMATHB");
public static final DefaultInputCharSetEnum wpextendedmatha = new Default
InputCharSetEnum("WPEXTENDEDMATHA");
public static final DefaultInputCharSetEnum wpextendedmathb = new Default
InputCharSetEnum("WPEXTENDEDMATHB");
public static final DefaultInputCharSetEnum wpmultinationala = new Default
InputCharSetEnum("WPMULTINATIONALA");
public static final DefaultInputCharSetEnum wpmultinationalb = new Default
InputCharSetEnum("WPMULTINATIONALB");
public static final DefaultInputCharSetEnum wpphonetic = new Default
InputCharSetEnum("WPPHONETIC");
public static final DefaultInputCharSetEnum wptypographic = new Default
InputCharSetEnum("WPTYPOGRAPHIC");
public static final DefaultInputCharSetEnum mtextra = new Default
InputCharSetEnum("MTEXTRA");
public static final DefaultInputCharSetEnum bookshelfsymbol3 = new Default
InputCharSetEnum("BOOKSHELFSYMBOL3");
public static final DefaultInputCharSetEnum hproman8 = new Default
InputCharSetEnum("HPROMAN8");
public static final DefaultInputCharSetEnum bidi_oldcode = new Default
InputCharSetEnum("BIDI-OLDCODE");
public static final DefaultInputCharSetEnum bidi_pc8 = new Default
InputCharSetEnum("BIDI-PC8");
public static final DefaultInputCharSetEnum bidi_e0 = new Default
InputCharSetEnum("BIDI-E0");
public static final DefaultInputCharSetEnum htmlkoi8 = new Default
InputCharSetEnum("HTMLKOI8");
public static final DefaultInputCharSetEnum jis_roman = new Default
InputCharSetEnum("JIS-ROMAN");
public static final DefaultInputCharSetEnum utf8 = new Default
```

```
InputCharSetEnum("UTF8");
public static final DefaultInputCharSetEnum utf7 = new Default
InputCharSetEnum("UTF7");
```

### FallbackFormatEnum

Valid for the fallbackFormat option. The class FallbackFormatEnum defines the following static members:

```
public static final FallbackFormatEnum ANSI_7            = new
FallbackFormatEnum("ANSI-7");
public static final FallbackFormatEnum ANSI_8            = new
FallbackFormatEnum("ANSI-8");
public static final FallbackFormatEnum ASCII_7           = new
FallbackFormatEnum("ASCII-7");
public static final FallbackFormatEnum ASCII_8           = new
FallbackFormatEnum("ASCII-8");
public static final FallbackFormatEnum BIG5              = new
FallbackFormatEnum("Big5");
public static final FallbackFormatEnum EUC_JP            = new
FallbackFormatEnum("EUC-JP");
public static final FallbackFormatEnum EUC_KR            = new
FallbackFormatEnum("EUC-KR");
public static final FallbackFormatEnum GB2312            = new
FallbackFormatEnum("GB2312");
public static final FallbackFormatEnum HEBREW_OLD_CODE   = new
FallbackFormatEnum("hebrew-old-code");
public static final FallbackFormatEnum ISO_10646_UCS_2   = new
FallbackFormatEnum("ISO-10646-UCS-2");
public static final FallbackFormatEnum ISO_2022_JP       = new
FallbackFormatEnum("ISO-2022-JP");
public static final FallbackFormatEnum ISO_8859_2        = new
FallbackFormatEnum("ISO-8859-2");
public static final FallbackFormatEnum ISO_8859_6        = new
FallbackFormatEnum("ISO-8859-2");
public static final FallbackFormatEnum KOI8_R            = new
FallbackFormatEnum("KOI8-R");
public static final FallbackFormatEnum SHIFT_JIS         = new
FallbackFormatEnum("Shift_JIS");
public static final FallbackFormatEnum UTF_8             = new
FallbackFormatEnum("UTF-8");
public static final FallbackFormatEnum WINDOWS_1250      = new
FallbackFormatEnum("windows-1250");
public static final FallbackFormatEnum WINDOWS_1251      = new
FallbackFormatEnum("windows-1251");
public static final FallbackFormatEnum WINDOWS_1252      = new
FallbackFormatEnum("windows-1252");
public static final FallbackFormatEnum WINDOWS_1253      = new
FallbackFormatEnum("windows-1253");
public static final FallbackFormatEnum WINDOWS_1254      = new
FallbackFormatEnum("windows-1254");
public static final FallbackFormatEnum WINDOWS_1255      = new
FallbackFormatEnum("windows-1255");
public static final FallbackFormatEnum WINDOWS_1256      = new
FallbackFormatEnum("windows-1256");
public static final FallbackFormatEnum WINDOWS_1257      = new
FallbackFormatEnum("windows-1257");
public static final FallbackFormatEnum WINDOWS_874       = new
FallbackFormatEnum("windows-874");
public static final FallbackFormatEnum X_MAC_ROMAN_7     = new
FallbackFormatEnum("x-Mac-roman-7");
public static final FallbackFormatEnum X_MAC_ROMAN       = new
FallbackFormatEnum("x-Mac-roman");
public static final FallbackFormatEnum NO_FALLBACK_FORMAT = new
FallbackFormatEnum("noFallbackFormat");
```

### DocumentMemoryModeEnum

Valid for the documentMemoryMode option. The class DocumentMemoryModeEnum defines the following static members:

```
public static final DocumentMemoryModeEnum SMALLESTMODE = new
DocumentMemoryModeEnum("smallestmode");
public static final DocumentMemoryModeEnum SMALLMODE    = new
DocumentMemoryModeEnum("smallmode");
public static final DocumentMemoryModeEnum MEDIUMMODE   = new
DocumentMemoryModeEnum("mediummode");
public static final DocumentMemoryModeEnum LARGEMODE    = new
DocumentMemoryModeEnum("largemode");
public static final DocumentMemoryModeEnum LARGESTMODE  = new
DocumentMemoryModeEnum("largestmode");
```

## HTML Export

This information applies to HTML Export.

### CharacterByteOrderEnum

Valid for the characterByteOrder option. The class CharacterByteOrderEnum defines the following static members:

```
public static final CharacterByteOrderEnum BIG_ENDIAN = new
CharacterByteOrderEnum("big-endian");
public static final CharacterByteOrderEnum LITTLE_ENDIAN = new
CharacterByteOrderEnum("little-endian");
public static final CharacterByteOrderEnum TEMPLATE_ORDER = new
CharacterByteOrderEnum("template-order");
```

### CharacterSetEnum

Valid for the outputCharacterSet option. The class CharacterSetEnum defines the following static members:

```
public static final CharacterSetEnum ISO_8859_1        = new
CharacterSetEnum("ISO-8859-1");
public static final CharacterSetEnum ISO_8859_2        = new
CharacterSetEnum("ISO-8859-2");
public static final CharacterSetEnum ISO_8859_3        = new
CharacterSetEnum("ISO-8859-3");
public static final CharacterSetEnum ISO_8859_4        = new
CharacterSetEnum("ISO-8859-4");
public static final CharacterSetEnum ISO_8859_5        = new
CharacterSetEnum("ISO-8859-5");
public static final CharacterSetEnum ISO_8859_6        = new
CharacterSetEnum("ISO-8859-6");
public static final CharacterSetEnum ISO_8859_7        = new
CharacterSetEnum("ISO-8859-7");
public static final CharacterSetEnum ISO_8859_8        = new
CharacterSetEnum("ISO-8859-8");
public static final CharacterSetEnum ISO_8859_9        = new
CharacterSetEnum("ISO-8859-9");
public static final CharacterSetEnum X_MAC_ROMAN       = new
CharacterSetEnum("x-Mac-roman");
public static final CharacterSetEnum X_MAC_CE          = new
CharacterSetEnum("x-Mac-ce");
public static final CharacterSetEnum X_MAC_GREEK       = new
CharacterSetEnum("x-Mac-Greek");
public static final CharacterSetEnum X_MAC_CYRILLIC    = new
CharacterSetEnum("x-Mac-Cyrillic");
public static final CharacterSetEnum X_MAC_TURKISH     = new
CharacterSetEnum("x-Mac-Turkish");
public static final CharacterSetEnum GB2312            = new
```

```
CharacterSetEnum("GB2312");
public static final CharacterSetEnum BIG5              = new
CharacterSetEnum("Big5");
public static final CharacterSetEnum SHIFT_JIS         = new
CharacterSetEnum("Shift_JIS");
public static final CharacterSetEnum KOI8_R            = new
CharacterSetEnum("KOI8-R");
public static final CharacterSetEnum WINDOWS_1250      = new
CharacterSetEnum("windows-1250");
public static final CharacterSetEnum WINDOWS_1251      = new
CharacterSetEnum("windows-1251");
public static final CharacterSetEnum WINDOWS_1252      = new
CharacterSetEnum("windows-1252");
public static final CharacterSetEnum WINDOWS_1253      = new
CharacterSetEnum("windows-1253");
public static final CharacterSetEnum WINDOWS_1254      = new
CharacterSetEnum("windows-1254");
public static final CharacterSetEnum WINDOWS_1255      = new
CharacterSetEnum("windows-1255");
public static final CharacterSetEnum WINDOWS_1256      = new
CharacterSetEnum("windows-1256");
public static final CharacterSetEnum WINDOWS_1257      = new
CharacterSetEnum("windows-1257");
public static final CharacterSetEnum EUC_KR            = new
CharacterSetEnum("EUC-KR");
public static final CharacterSetEnum EUC_JP            = new
CharacterSetEnum("EUC-JP");
public static final CharacterSetEnum ISO_2022_JP       = new
CharacterSetEnum("ISO-2022-JP");
public static final CharacterSetEnum WINDOWS_874       = new
CharacterSetEnum("windows-874");
public static final CharacterSetEnum UTF_7             = new
CharacterSetEnum("UTF-7");
public static final CharacterSetEnum UTF_8             = new
CharacterSetEnum("UTF-8");
public static final CharacterSetEnum ISO_10646_UCS_2   = new
CharacterSetEnum("ISO-10646-UCS-2");
public static final CharacterSetEnum X_CHARSET_UNKNOWN = new
CharacterSetEnum("x-Charset-Unknown");
```

### ComplianceEnum

Valid for the compliance option. The class ComplianceEnum defines the following static members:

```
public static final ComplianceEnum NONE        = new ComplianceEnum("none");
public static final ComplianceEnum WELL_FORMED = new ComplianceEnum("well-
formed");
public static final ComplianceEnum STRICT_DTD  = new
ComplianceEnum("strictDTD");
```

### ExtractEmbeddedFilesEnum

Valid for the extractEmbeddedFiles option. The class ExtractEmbeddedFilesEnum defines the following static members:

```
public static final ExtractEmbeddedFilesEnum IGNOREFILES   = new
ExtractEmbeddedFilesEnum("ignoreFiles");
public static final ExtractEmbeddedFilesEnum CONVERTFILES   = new
ExtractEmbeddedFilesEnum("convertFiles");
public static final ExtractEmbeddedFilesEnum EXTRACTFILES   = new
ExtractEmbeddedFilesEnum("extractFiles");
```

### FlavorEnum

Valid for the flavor option. The class FlavorEnum defines the following static members:

```
public static final FlavorEnum GENERIC_HTML            = new
FlavorEnum("generic-html");
public static final FlavorEnum GENERIC_WIRELESS_HTML   = new
FlavorEnum("generic-wireless-html");
public static final FlavorEnum HTML_20                 = new
FlavorEnum("html2.0");
public static final FlavorEnum HTML_30                 = new
FlavorEnum("html3.0");
public static final FlavorEnum HTML_40                 = new
FlavorEnum("html4.0");
public static final FlavorEnum NETSCAPE_30             = new
FlavorEnum("netscape3.0");
public static final FlavorEnum NETSCAPE_40             = new
FlavorEnum("netscape4.0");
public static final FlavorEnum IE_30                   = new
FlavorEnum("internetExplorer3.0");
public static final FlavorEnum IE_40                   = new
FlavorEnum("internetExplorer4.0");
public static final FlavorEnum AVANTGO_33_PALM         = new
FlavorEnum("avantGo3.3-palm");
public static final FlavorEnum AVANTGO_33_PALM_NOTBLS  = new
FlavorEnum("avantGo3.3-palm-noTables");
public static final FlavorEnum AVANTGO_33_WINCE        = new
FlavorEnum("avantGo3.3-winCE");
public static final FlavorEnum AVANTGO_33_WINCE_NOTBLS = new
FlavorEnum("avantGo3.3-winCE-noTables");
public static final FlavorEnum WEBCLIPPING_11          = new
FlavorEnum("webClipping1.1");
public static final FlavorEnum WEBCLIPPING_11_NOTBLS   = new
FlavorEnum("webClipping1.1-noTables");
public static final FlavorEnum CHTML_20                = new
FlavorEnum("chtml2.0");
public static final FlavorEnum HDML_30                 = new
FlavorEnum("hdml3.0");
public static final FlavorEnum TEXT                    = new FlavorEnum("text");
public static final FlavorEnum WML_11                  = new
FlavorEnum("wml1.1");
public static final FlavorEnum WML_11_WITHTBLS         = new FlavorEnum("wml1.1-
withTables");
public static final FlavorEnum WML_20                  = new
FlavorEnum("wml2.0");
public static final FlavorEnum XHTML_BASIC_10          = new FlavorEnum("xhtml-
basic1.0");
public static final FlavorEnum XHTML_BASIC_10_NOTBLS   = new FlavorEnum("xhtml-
basic1.0-noTables");
```

### GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The class GraphicSizeMethodEnum defines the following static members:

```
public static final GraphicSizeMethodEnum SMOOTH     = new
GraphicSizeMethodEnum("smooth");
public static final GraphicSizeMethodEnum QUICK      = new
GraphicSizeMethodEnum("quick");
public static final GraphicSizeMethodEnum SMOOTHGRAY = new
GraphicSizeMethodEnum("smoothGray");
```

### GraphicTypeEnum

Valid for the graphicType option. The class GraphicTypeEnum defines the following static members:

```
public static final GraphicTypeEnum BMP         = new GraphicTypeEnum("bmp");
public static final GraphicTypeEnum GIF         = new GraphicTypeEnum("gif");
public static final GraphicTypeEnum JPEG        = new GraphicTypeEnum("jpeg");
public static final GraphicTypeEnum NO_GRAPHICS = new
GraphicTypeEnum("noGraphics");
```

```
public static final GraphicTypeEnum PNG         = new GraphicTypeEnum("png");
public static final GraphicTypeEnum WBMP        = new GraphicTypeEnum("wbmp");
```

### GridAdvanceEnum

Valid for the gridAdvance option. The class GridAdvanceEnum defines the following static members:

```
public static final GridAdvanceEnum ADVANCE_ACROSS = new
GridAdvanceEnum("advanceAcross");
public static final GridAdvanceEnum ADVANCE_DOWN   = new
GridAdvanceEnum("advanceDown");
```

### ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
public static final ReorderMethodEnum OFF = new ReorderMethodEnum("reorderOff");
public static final ReorderMethodEnum RIGHTTOLEFT = new
ReorderMethodEnum("reorderRightToLeft");
public static final ReorderMethodEnum LEFTTORIGHT = new
ReorderMethodEnum("redorderLeftToRight");
```

### SpreadSheetBordersEnum

Valid for the spreadsheetBorders option. The class SpreadSheeteBordersEnum defines the following static members:

```
public static final SpreadsheetBordersEnum CREATEBORDERIFMISSING = new
SpreadsheetFitToPageEnum("createBorderIfMissing");
public static final SpreadsheetBordersEnum BORDERSOFF = new
SpreadsheetFitToPageEnum("bordersOff");
public static final SpreadsheetBordersEnum USESOURCEBORDERS = new
SpreadsheetFitToPageEnum("useSourceBorders")
```

## Search Export

This information applies to Search Export.

### OleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

```
public static final OleEmbeddingsEnum PROCESSSTANDARD = new
OleEmbeddingsEnum ("processStandard");
public static final OleEmbeddingsEnum PROCESSALL = new
OleEmbeddingsEnum ("processAll");
public static final OleEmbeddingsEnum PROCESSNONE = new
OleEmbeddingsEnum ("processNone");
```

### SearchMLUnmappedTextEnum

Valid for the unmappedText option. The class SearchMLUnmappedTextEnum defines the following static members:

```
public static final SearchMLUnmappedTextEnum JUSTUNMAPPEDTEXT = new
SearchMLUnmappedTextEnum("justUnmappedText");
public static final SearchMLUnmappedTextEnum NOUNMAPPEDTEXT = new
SearchMLUnmappedTextEnum("noUnmappedText");
public static final SearchMLUnmappedTextEnum BOTHUNMAPPEDTEXT = new
SearchMLUnmappedTextEnum("bothUnmappedText");
```

### XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option. The class XmlDefinitionMethodEnum defines the following static members:

```
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("dtd");
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("noDefinition");
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("xsd");
```

## Image Export

This information applies to Image Export.

### DatabaseFitToPageEnum

Valid for the databaseFitToPage option. The class DatabaseFitToPageEnum defines the following static members:

```
public static final DatabaseFitToPageEnum NO_SCALING   = new
DatabaseFitToPageEnum("dbNoScaling");
public static final DatabaseFitToPageEnum FIT_TO_PAGE  = new
DatabaseFitToPageEnum("dbFitToPage");
public static final DatabaseFitToPageEnum FIT_TO_WIDTH = new
DatabaseFitToPageEnum("dbFitToWidth");
public static final DatabaseFitToPageEnum FIT_TOHEIGHT = new
DatabaseFitToPageEnum("dbFitToHeight");
```

### GraphicCroppingEnum

Valid for the graphicCropping option. The class GraphicCroppingEnum defines the following static members:

```
public static final GraphicCroppingEnum NO_CROPPING    = new
GraphicCroppingEnum("noCropping");
public static final GraphicCroppingEnum CROP_TO_CONTENT= new
GraphicCropgingEnum("cropToContent");
```

### GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The class GraphicSizeMethodEnum defines the following static members:

```
public static final GraphicSizeMethodEnum SMOOTH     = new
GraphicSizeMethodEnum("smooth");
public static final GraphicSizeMethodEnum QUICK      = new
GraphicSizeMethodEnum("quick");
public static final GraphicSizeMethodEnum SMOOTHGRAY = new
GraphicSizeMethodEnum("smoothGray");
```

### GraphicWatermarkScaleTypeEnum

Valid for the graphicWatermarkScaleType option. The class GraphicWatermarkScaleTypeEnum defines the following static members:

```
public static final GraphicWatermarkScaleTypeEnum SCALEWATERMARKOFF = new
GraphicWatermarkScaleTypeEnum("scaleWatermarkOff");
public static final GraphicWatermarkScaleTypeEnum SCALEWATERMARKBYPERCENT = new
GraphicWatermarkScaleTypeEnum("scaleWatermarkByPercent");
```

### MimeHeaderOutputEnum

Valid for the mimeHeaderOutput option. The class MimeHeaderOutputEnum defines the following static members:

```
public static final MimeHeaderOutputEnum ALL      = new
MimeHeaderOutputEnum("string")
public static final MimeHeaderOutputEnum STANDARD = new
MimeHeaderOutputEnum("standard")
```

### ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
public static final ReorderMethodEnum OFF = new ReorderMethodEnum("reorderOff");
public static final ReorderMethodEnum RIGHTTOLEFT = new
ReorderMethodEnum("reorderRightToLeft");
public static final ReorderMethodEnum LEFTTORIGHT = new
ReorderMethodEnum("redorderLeftToRight");
```

### SpreadsheetFitToPageEnum

Valid for the spreadsheetFitToPage option. The class SpreadsheetFitToPageEnum defines the following static members:

```
public static final SpreadsheetFitToPageEnum NO_SCALING          = new
SpreadsheetFitToPageEnum("ssNoScaling");
public static final SpreadsheetFitToPageEnum FIT_TO_PAGE         = new
SpreadsheetFitToPageEnum("ssFitToPage");
public static final SpreadsheetFitToPageEnum FIT_TO_WIDTH        = new
SpreadsheetFitToPageEnum("ssFitToWidth");
public static final SpreadsheetFitToPageEnum FIT_TO_HEIGHT       = new
SpreadsheetFitToPageEnum("ssFitToHeight");
public static final SpreadsheetFitToPageEnum SCALE_BY_PERCENTAGE = new
SpreadsheetFitToPageEnum("ssScaleByPercentage");
public static final SpreadsheetFitToPageEnum FIT_TO_PAGES        = new
SpreadsheetFitToPageEnum("ssFitToPages");
```

### SpreadsheetPageDirectionEnum

Valid for the spreadsheetPageDirection option. The class SpreadsheetPageDirectionEnum defines the following static members:

```
public static final SpreadsheetPageDirectionEnum DOWN_THEN_ACROSS = new
SpreadsheetPageDirectionEnum("downThenAcross");
public static final SpreadsheetPageDirectionEnum ACROSS_THEN_DOWN = new
SpreadsheetPageDirectionEnum("acrossThenDown");
```

### TiffByteOrderEnum

Part of the TiffOptions structure. The class TiffByteOrderEnum defines the following static members:

```
public static final TiffFillOrderEnum LITTLE-ENDIAN = new
TiffFillOrderEnum("little-endian");
public static final TiffFillOrderEnum BIG-ENDIAN = new TiffFillOrderEnum("big-
endian");
```

### TiffColorSpaceEnum

Part of the TiffOptions structure. The class TiffColorSpaceEnum defines the following static members:

```
public static final TiffColorSpaceEnum BLACKWHITE_1BIT = new
TiffColorSpaceEnum("blackWhite-1Bit");
public static final TiffColorSpaceEnum PALETTE_8BIT    = new
TiffColorSpaceEnum("palette-8Bit");
public static final TiffColorSpaceEnum RGB_24BIT       = new
TiffColorSpaceEnum("rgb-24Bit");
```

### TiffCompressionEnum

Part of the TiffOptions class. The class TiffCompressionEnum defines the following static members:

```
public static final TiffCompressionEnum NO_COMPRESSION    = new
TiffCompressionEnum("noCompression");
```

```
public static final TiffCompressionEnum PACKBITS         = new
TiffCompressionEnum("packbits");
public static final TiffCompressionEnum LZW              = new
TiffCompressionEnum("LZW");
public static final TiffCompressionEnum CCITT_1D         = new
TiffCompressionEnum("CCITT-1D");
public static final TiffCompressionEnum CCITT_GROUP3_FAX  = new
TiffCompressionEnum("CCITT-Group3-Fax");
public static final TiffCompressionEnum CCITT_GROUP43_FAX = new
TiffCompressionEnum("CCITT-Group4-Fax");
```

### TiffFillOrderEnum

Part of the TiffOptions structure. The class TiffFillOrderEnum defines the following static members:

```
public static final TiffFillOrderEnum FILLORDER-1 = new
TiffFillOrderEnum("fillOrder-1");
public static final TiffFillOrderEnum FILLORDER-2 = new
TiffFillOrderEnum("fillOrder-2");
```

## PDF Export

This information applies to PDF Export.

### DefaultPageUnitsEnum

Valid for the defaultPageUnits option. The class DefaultPageUnitsEnum defines the following static members:

```
public static final DefaultPageUnitsEnum INCHES   = new
DefaultPageUnitsEnum("INCHES");
public static final DefaultPageUnitsEnum POINTS   = new
DefaultPageUnitsEnum("POINTS");
public static final DefaultPageUnitsEnum CENTIMETERS   = new
DefaultPageUnitsEnum("CENTIMETERS");
public static final DefaultPageUnitsEnum PICAS   = new
DefaultPageUnitsEnum("PICAS");
} OIT_DefaultPageUnitsEnum;
```

### ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
public static final ReorderMethodEnum OFF = new ReorderMethodEnum("reorderOff");
public static final ReorderMethodEnum RIGHTTOLEFT = new
ReorderMethodEnum("reorderRightToLeft");
public static final ReorderMethodEnum LEFTTORIGHT = new
ReorderMethodEnum("redorderLeftToRight");
```

### WatermarkPositionEnum

Valid for the watermarkPosition option. The class WatermarkPositionEnum defines the following static members:

```
public static final WatermarkPositionEnum CENTEROFPAGE = new
WatermarkPositionEnum("centerOfPage");
```

### WatermarkScalingEnum

Valid for the watermarkScaling option. The class WatermarkScalingEnum defines the following static members:

```
public static final WatermarkScalingEnum PDFNOMAP = new
WatermarkScalingnEnum("pdfNoMap");
public static final WatermarkScalingEnum PDFFITTOPAGE = new
WatermarkScalingnEnum("pdfFitToPage");
```

```
public static final WatermarkScalingEnum PDFSCALE = new
WatermarkScalingnEnum("pdfScale");
```

## XML Export

This information applies to XML Export.

### GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The class GraphicSizeMethodEnum defines the following static members:

```
public static final GraphicSizeMethodEnum SMOOTH     = new
GraphicSizeMethodEnum("smooth");
public static final GraphicSizeMethodEnum QUICK      = new
GraphicSizeMethodEnum("quick");
public static final GraphicSizeMethodEnum SMOOTHGRAY = new
GraphicSizeMethodEnum("smoothGray");
```

### GraphicTypeEnum

Valid for the graphicType option. The class GraphicTypeEnum defines the following static members:

```
public static final GraphicTypeEnum BMP         = new GraphicTypeEnum("bmp");
public static final GraphicTypeEnum GIF         = new GraphicTypeEnum("gif");
public static final GraphicTypeEnum JPEG        = new GraphicTypeEnum("jpeg");
public static final GraphicTypeEnum NO_GRAPHICS = new
GraphicTypeEnum("noGraphics");
public static final GraphicTypeEnum PNG         = new GraphicTypeEnum("png");
public static final GraphicTypeEnum WBMP        = new GraphicTypeEnum("wbmp");
```

### OleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

```
public static final OleEmbeddingsEnum PROCESSSTANDARD = new
OleEmbeddingsEnum ("processStandard");
public static final OleEmbeddingsEnum PROCESSALL = new
OleEmbeddingsEnum ("processAll");
public static final OleEmbeddingsEnum PROCESSNONE = new
OleEmbeddingsEnum ("processNone");
```

### ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

```
public static final ReorderMethodEnum OFF = new
ReorderMethodEnum("reorderOff");
public static final ReorderMethodEnum RIGHTTOLEFT = new
ReorderMethodEnum("reorderRightToLeft");
public static final ReorderMethodEnum LEFTTORIGHT = new
ReorderMethodEnum("redorderLeftToRight");
```

### XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option. The class XmlDefinitionMethodEnum defines the following static members:

```
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("dtd");
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("noDefinition");
public static final XmlDefinitionMethodEnum DTD = new
XmlDefinitionMethodEnum("xsd");
```

# D

# Copyrights and Licensing

This appendix provides a comprehensive overview of all copyright and licensing information for Outside In Transformation Server.

## Outside In Transformation Server Licensing

This product includes code licensed from RSA Data Security.

Portions relating to XServer copyright 1990, 1991 Network Computing Devices, 1987 Digital Equipment Corporation and the Massachusetts Institute of Technology.Portions of this software are copyright © 1996-2002 The FreeType Project (www.freetype.org). All rights reserved.Portions copyright 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Cold Spring Harbor Laboratory. Funded under Grant P41-RR02188 by the National Institutes of Health. Portions copyright 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Boutell.Com, Inc. Portions relating to GD2 format copyright 1999, 2000, 2001, 2002 Philip Warner. Portions relating to PNG copyright 1999, 2000, 2001, 2002 Greg Roelofs. Portions relating to PNG Copyright 1995-1996 Jean-loup Gailly and Mark Adler

Portions relating to PNG Copyright 1998, 1999 Glenn Randers-Pehrson, Tom Lane, Willem van Schaik, John Bowler, Kevin Bracey, Sam Bushell, Magnus Holmgren, Greg Roelofs, Tom Tanner, Andreas Dilger, Dave Martindale, Guy Eric Schalnat, Paul Schmidt, Tim WegnerPortions relating to gdttf.c copyright 1999, 2000, 2001, 2002 John Ellson (ellson@graphviz.org). Portions relating to gdft.c copyright 2001, 2002 John Ellson (ellson@graphviz.org). Portions relating to JPEG and to color quantization copyright 2000, 2001, 2002, Doug Becker and copyright (C) 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, Thomas G. Lane. This software is based in part on the work of the Independent JPEG Group. See the file README-JPEG.TXT for more information. Portions relating to WBMP copyright 2000, 2001, 2002 Maurice Szmurlo and Johan Van den Brande. Portions relating to GIF Copyright 1987, by Steven A. Bennett.

Permission has been granted to copy, distribute and modify gd in any context without fee, including a commercial application, provided that this notice is present in user-accessible supporting documentation. This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for the authors of gd, not to interfere with your productive use of gd. If you have questions, ask. "Derived works" includes all programs that utilize the library. Credit must be given in user-accessible documentation. This software is provided "AS IS." The copyright holders disclaim all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to this code and accompanying documentation. Although their code does not appear in gd 2.0.4, the authors wish to thank David Koblas, David Rowley, and Hutchison Avenue Software Corporation for their prior contributions.

BSD License for kXML2Copyright (c) 2002,2003, Stefan Haustein, Oberhausen, Rhld., GermanyPermission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANYCLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following copyrights are the properties of their respective owners: This product includes software developed by the Apache Software Foundation (http://

www.apache.org/). Copyright © 1999 The Apache Software Foundation. All rights reserved. UnRAR - free utility for RAR archivesLicense for use and distribution of FREE portable versionThe source code of UnRAR utility is freeware. This means:1. All copyrights to RAR and the utility UnRAR are exclusively owned by the author - Alexander Roshal.2. The UnRAR sources may be used in any software to handle RAR archives without limitations free of charge, but cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified UnRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver.3. The UnRAR utility may be freely distributed. No person or company may charge a fee for the distribution of UnRAR without written permission from the copyright holder.4. THE RAR ARCHIVER AND THE UNRAR UTILITY ARE DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.5. Installing and using the UnRAR utility signifies acceptance of these terms and conditions of the license.6. If you don't agree with terms of the license you must remove UnRAR files from your storage devices and cease to use the utility.

# Index

## F



## G



## H



## I



## J

## L



## M



## N



## O



## P



## R

## S



## T



## U



## V

## W



## X