Oracle® Outside In Transformation Server

Developer's Guide

Release 8.4.1

E12868-05

May 2013

ORACLE®

Oracle Outside In Transformation Server Developer's Guide, Release 8.4.1

E12868-05

Primary Author: Mike Manier

Contents

6 Transformation Engine Specification

7 IO Provider Specification

B C/C++ Client Data Types

C Java Client Data Types

Preface

This document covers the installation and use of the Outside In Transformation Server.

Audience

This document is intended for developers who are investigating the 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 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.

Convention	Meaning
`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?

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:

1.1 What's New in Release 8.4.1

- 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 to identify DICOM (Digital Imaging and Communications in Medicine) files.
- The following Microsoft Office formats are now supported: Microsoft Word 2011 for Mac, Microsoft Excel 2011 for Mac, Microsoft PowerPoint 2011 for Mac, Microsoft Word 2013, Microsoft Excel 2013, Microsoft PowerPoint 2013, Microsoft Outlook 2013.
- The following Adobe Creative Suite formats are now supported: Photoshop CS6, Illustrator CS6, InDesign CS6.
- Support has been added for Windows 8 and Windows 2012 Server on the Windows x86-64 platform.

1.2 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.

1.2.1 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 throught the Transformation Manager.

1.2.2 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.

1.2.3 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.

1.2.4 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.

1.3 Architecture

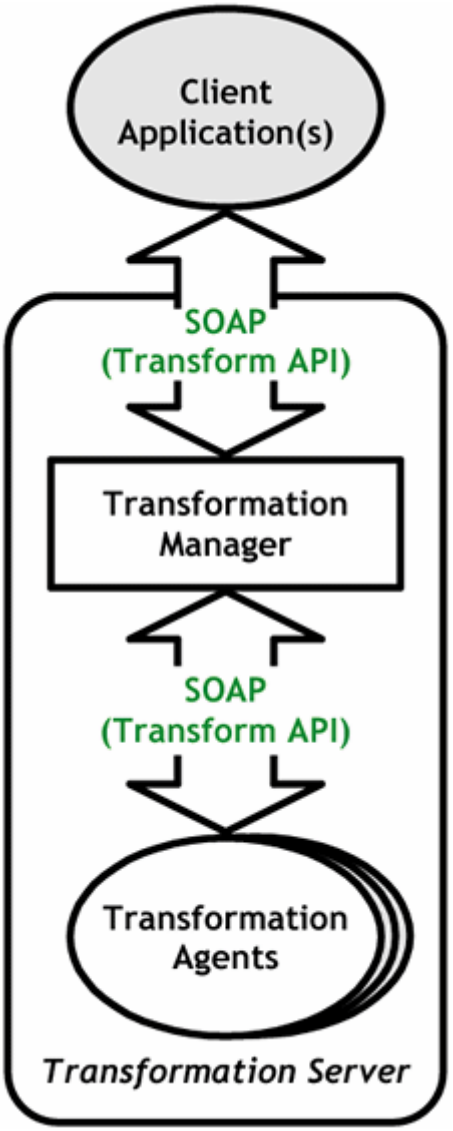
This section describes the system software architecture.

1.3.1 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 Section 1.3.2, "The Transformation Agent").

The following ilustration depicts this communication.

Figure 1–1 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.

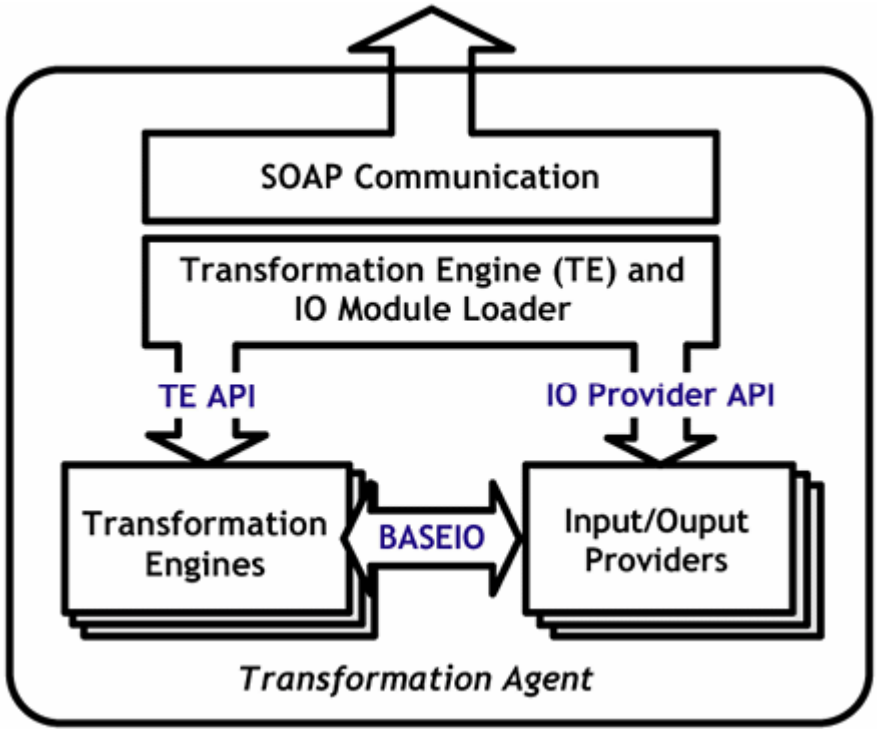
The Transform API is accessed via SOAP messages, through C-language function calls using the C Client Module, and /or through Java methods presented by the Java Client object.

1.3.2 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 1–2 Transformation Agent



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

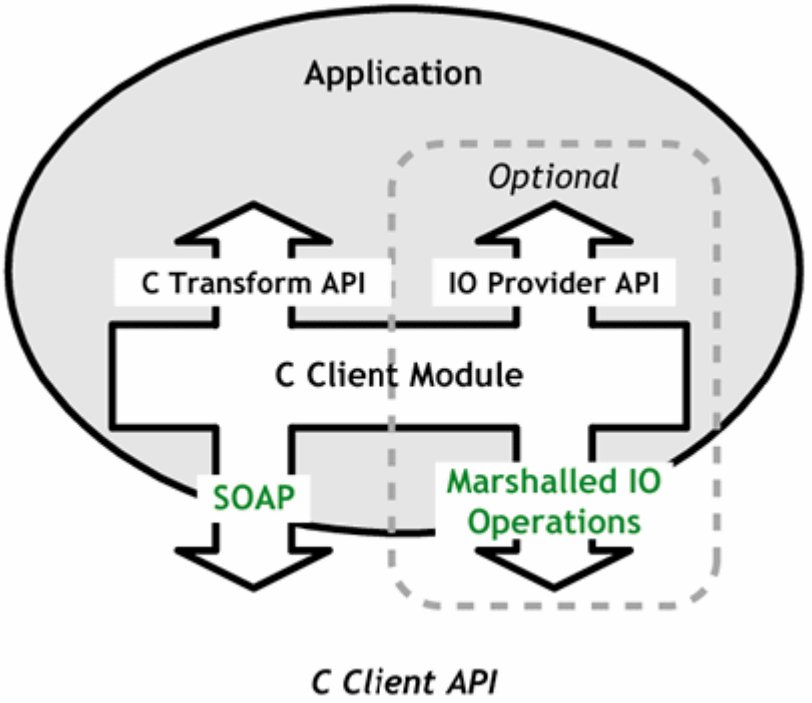
The C language interfaces between the Transformation Agent, the Transformation Engines, and IO Providers are fully documented in this guide.

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.

1.3.3 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 1–3 IO Provider API



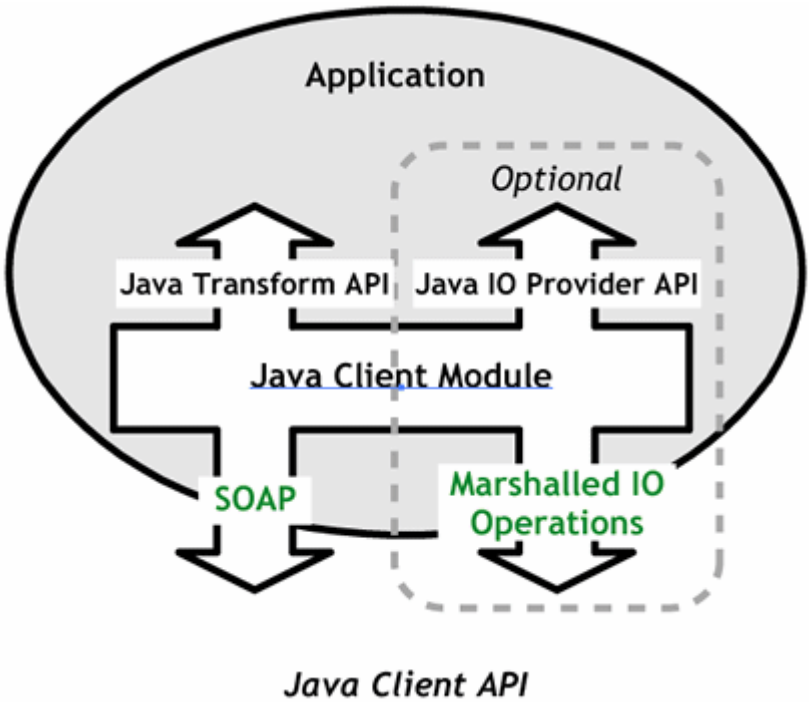
If an application provides custom IO on the client side, the interface module will marshal IO operations between the client process and Transformation Agent, over a socket connection.

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.

1.3.4 Java Client Object

Much like the C/C++ client module, the Java client object provides a Java version of the Transform API (see Section 5, "Initiating Transformations Using the Java API") that allows an application to access Transformation Server directly through code, rather than through the SOAP protocol.

Figure 1–4 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.)

1.4 Directory Structure

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

- **docs** - Contains both a PDF and HTML version of the product manual.
- **redist** - Contains only the files that the customer is allowed to redistribute. These include all the compiled modules, filter support files, .xsd and .dtd files, cmmap000.bin, and third-party libraries, 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.

1.5 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, 2013 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:

2.1 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.

Important: 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.

2.1.1 Installing Multiple SDKs

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

- **docs** – copy all subdirectories named "[product name]guide" into this directory.
- **redist** – copy all binaries into this directory.

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

2.1.2 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

2.1.3 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.

2.1.4 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
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

File	Description
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

2.2 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

2.2.1 tsmanager

The following is an overview of product details.

2.2.1.1 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.

2.2.1.2 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).

2.2.1.2.1 --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>.

2.2.1.2.2 --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>.

2.2.1.2.3 --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 <pipDir> subelement of <logInfo>.

2.2.1.2.4 --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>.

2.2.1.3 Command Line Flags

The following command line flags are used.

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

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

2.2.1.4 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 Section 2.2.1.2, "Command Line Options with Parameters."
- loghostname: Logs the client host name (slower than logip).
- logip: Logs the client's IP address.
- numagents: Described in Section 2.2.1.2, "Command Line Options with Parameters."
- pipedir: Described in Section 2.2.1.2, "Command Line Options with Parameters."
- port: Described in Section 2.2.1.2, "Command Line Options with Parameters."
- trace_on: Described in Section 2.2.1.3, "Command Line Flags."
- version: Described in Section 2.2.1.3, "Command Line Flags"

2.2.1.5 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
```

2.2.1.6 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.

2.2.1.7 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>
```

2.2.2 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.

2.2.2.1 Command Line Flags with Parameters

The following are command line flags with parameters.

2.2.2.1.1 --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.

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

2.2.2.2 Command Line Flags

The following are other command line flags.

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

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

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

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

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

2.3 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.

2.3.1 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>
```

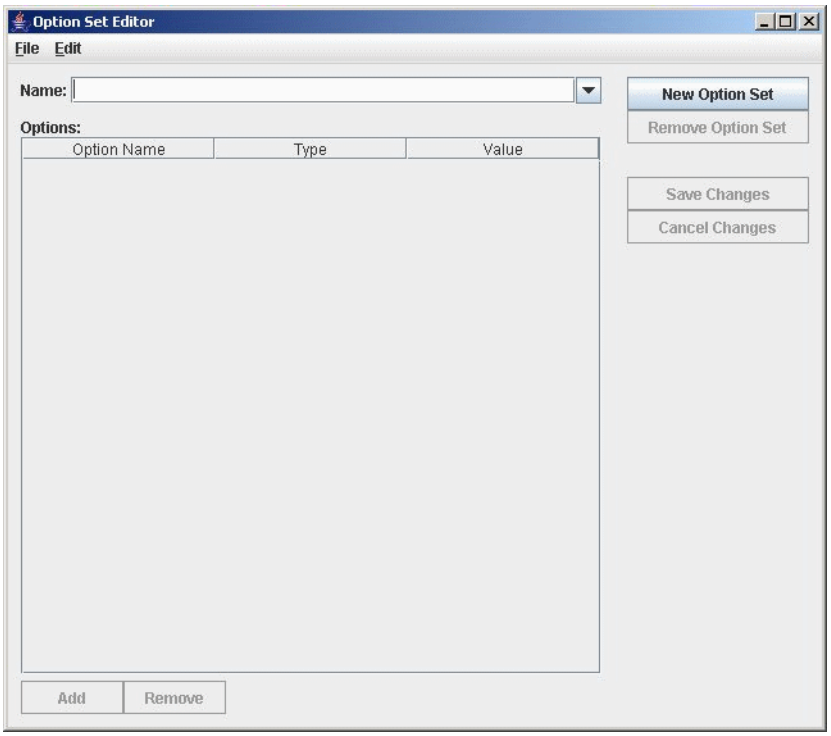
2.4 The Option Set Editor

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.

2.4.1 Using the Option Set Editor

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

Figure 2–1 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.

2.5 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 Chapter 6, "Transformation Engine Specification" for details.
- The IO Provider specification allows your application to extend the input and output functionality beyond the file system. See Chapter 7, "IO Provider Specification" for details.

3

Initiating Transformations Using 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:

3.1 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).

3.2 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.

3.3 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.

3.3.1 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).

3.3.2 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.

3.3.3 Example

The following are examples to demonstrate the interface.

3.3.3.1 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
```

3.3.3.2 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>
```

3.3.3.3 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.

3.3.4 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.

3.3.4.1 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.

3.3.4.2 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

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:

4.1 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).

4.1.1 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 Section 8.5.1, "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 */
}
```

4.2 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.

4.3 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 Appendix B, "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.

4.4 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.

4.5 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 Section 8.4.1, "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 Section 8.4.1, "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 Section 2.4, "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 Section 4.2, "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.

4.6 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).

4.7 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
  ```

4.7.1 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.

4.7.2 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 [IOServer] OutputFile ConfigurationFile [IO
Type]
```

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

5

Initiating Transformations Using 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 1.4.1 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:

5.1 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.

5.2 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.

5.3 Redirected IO

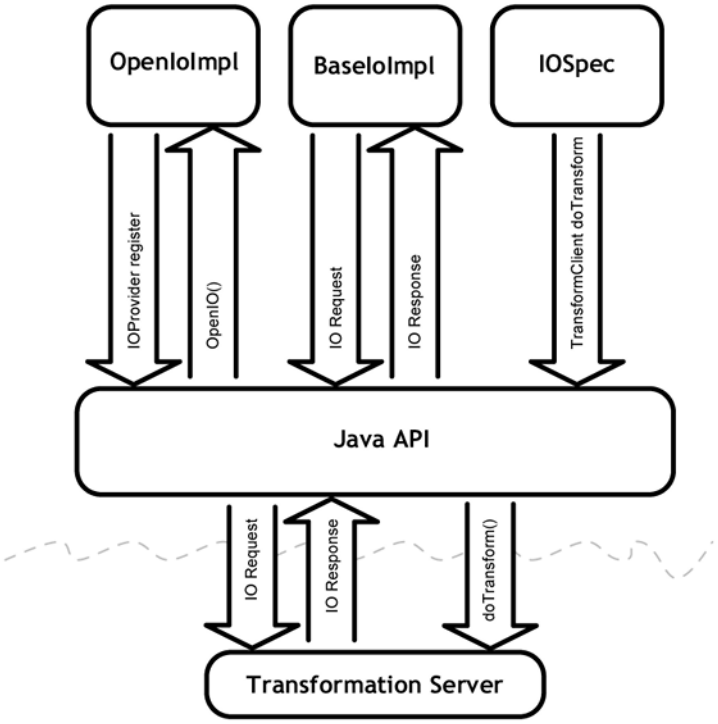
Transformation Server allows developers to specify how IO should be performed. Currently, Transformation Server handles three types of IO natively; file system, URL and redirected IO. When specifying the path using the type path, Transformation Server treats the file specification as a path to a file 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 5–1 Java API



5.4 Sample Applications

The folowing sections contains sample applications.

5.4.1 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.

5.4.1.1 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.

5.4.2 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.

5.4.3 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

A Transformation Engine is a module that is 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:

6.1 Getting Started

The following section describe basic steps in getting started.

6.1.1 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.

6.1.1.1 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.

6.1.1.2 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 Section 6.3, "Engine Interface" for details.

6.1.1.3 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 Section 6.4, "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.

6.1.2 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.

6.1.3 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.

6.2 Transformation Engine Entry Point

The following is entry point information.

6.2.1 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.

6.3 Engine Interface

The following sections describe the engine interface.

6.3.1 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

6.3.2 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.

6.3.3 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.

6.3.4 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.

6.3.5 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.

6.4 Agent Interface

The following information pertains to agent interface features.

6.4.1 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

6.4.2 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.

6.4.3 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

6.4.4 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

6.4.5 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

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 Section 5.3, "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:

7.1 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.

7.1.1 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.

7.1.2 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.

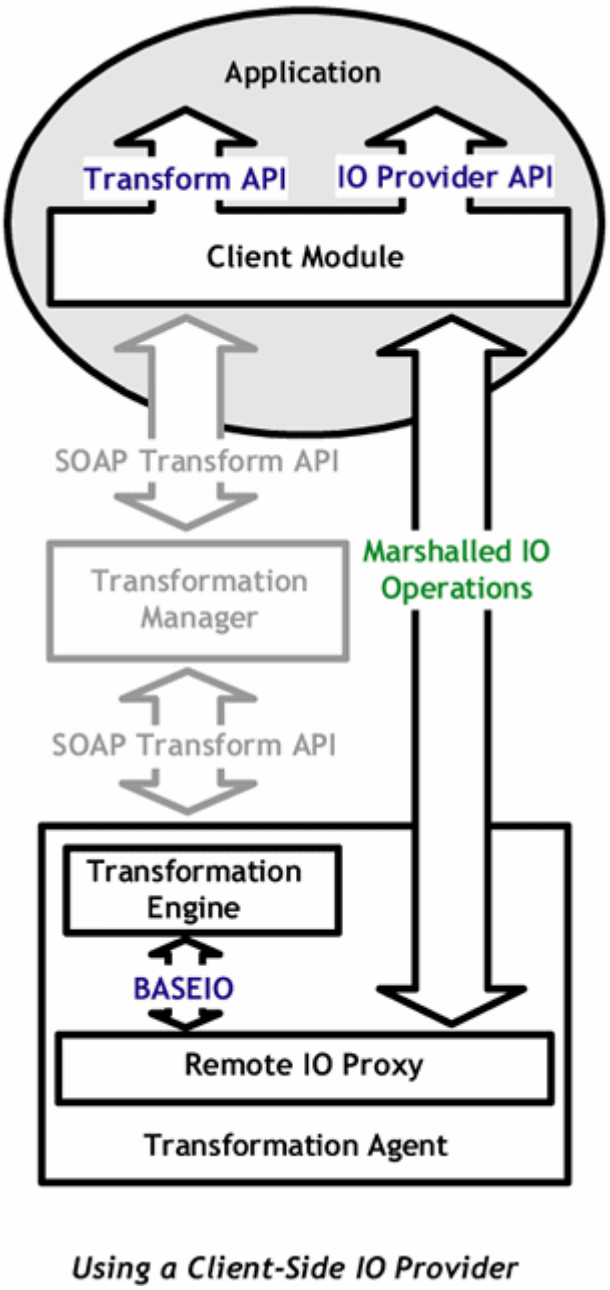
7.1.3 Server-Side Versus Client-Side IO Providers

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 7–1 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.

7.1.4 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.

7.1.5 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.

7.2 Configuration

This section contains basic configuration information.

7.2.1 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.

7.2.1.1 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>
```

7.2.1.2 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.

7.3 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.

7.3.1 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.

7.3.2 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;
```

7.4 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.

7.4.1 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.

7.4.2 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.

7.4.3 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.

7.4.4 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.

7.4.5 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.

7.4.6 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.

7.4.6.1 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.

7.4.6.1.1 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 );
```

7.4.6.1.2 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 );
```

7.4.6.1.3 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 );
```

7.4.6.1.4 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.

7.4.6.1.5 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.

7.4.6.1.6 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.

7.5 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.

7.5.1 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.

7.5.2 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

7.5.3 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.

7.5.4 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.

7.5.5 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 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 document assumes that the reader is familiar with the Outside In API.

This chapter includes the following sections:

8.1 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.

8.2 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.

8.3 Setting Transformation Parameters

This section describes transformation parameters.

8.3.1 Options

The following information concerns options.

8.3.1.1 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);
```

8.3.1.2 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));
```

8.3.1.3 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));
```

8.3.2 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.

8.3.2.1 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.

8.3.2.2 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.

8.3.2.3 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);
```

8.3.2.4 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) );
```

8.3.2.5 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

8.4 Performing a Transformation

This information pertains to performing transformations.

8.4.1 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";
```

8.4.2 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);
}
```

8.4.3 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;
```

8.5 Advanced Transformation Operations

This section pertains to advanced transformation options.

8.5.1 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.

8.5.1.1 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.

8.5.1.2 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.

8.5.1.2.1 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.

8.5.1.2.2 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.

8.5.1.3 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.

8.5.1.4 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.

8.6 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.

8.6.1 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: charMappingBoth charMappingText charMappingNone charMappingDefault convertChartObjects convertDateTimeProperties convertImageObjects convertPresentationObjects convertVectorObjects delimiters flattenStyles generateSystemData noBitmapElements noChartElements noPresentationElements noVectorElements separateStyleTables 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

Embedded API	SOAP API
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_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: readBufferSize memoryMappedInputSize 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

8.6.2 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

Embedded API	SOAP API
SCCOPT_DEFAULTPAGESIZE	The functionality of this option is supported by three options in the server implementation: defaultPageUnits defaultPageHeight defaultPageWidth
SCCOPT_DOLINEARIZATION	doLinearization
SCCOPT_EMBEDFONTS	embedFonts
SCCOPT_EX_UNICODECALLBACKSTR	NA
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: excludeFont 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: readBufferSize memoryMappedINputSize 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

Embedded API	SOAP API
SCCOPT_UNMAPPABLECHAR	unmappableCharacter
SCCOPT_USEDOCPAGESETTINGS	useDocumentPageSettings
SCCOPT_WATERMARKIO	The functionality of this option is supported by three options in the server implementation: watermarkImage watermarkScaling watermarkScalePercent
SCCOPT_WATERMARKPOSITION	watermarkPosition
SCCOPT_WHATTOPRINT	usePageRange

8.6.3 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

Embedded API	SOAP API
SCCOPT_IO_BUFFERSIZE	Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API: readBufferSize memoryMappedInputSize 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

8.6.4 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

Embedded API	SOAP API
SCCOPT_IO_BUFFERSIZE	Each of the flags in the embedded option has a matching, stand-alone option in the SOAP API: readBufferSize memoryMappedInputSize 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: textOutOn 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: allCapsOn boldOn doubleUnderlineOn hiddenOn italicOn originalCharsetOn outlineOn revisionAddOn revisionDeleteOn smallCapsOn strikeoutOn underlineOn
SCCOPT_XML_SEARCHML_FLAGS	Each of the flags in the embedded option has a matching, stand-alone Boolean option in the SOAP API: cellInfoOn changeNumbertoTextOn documentPropertiesOn embeddingsOn errorInfoOn generateSystemData metadataOnlyOn paragraphStyleNamesOn produceURLsOn revisionsOn suppressArchiveSubDocsOn suppressAttachmentsOn xmlDeclarationOff
SCCOPT_XML_SEARCHML_OFFSET	includeTextOffsets
SCCOPT_XML_SEARCHML_PARA_ATTR	paragraphAttributes
SCCOPT_XML_SEARCHML_UNMAPPEDTEXT	unmappedText

8.6.5 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
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

Embedded API	SOAP API
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: readBuffermemory MappedInputSize 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

All options discussed in this appendix are describedin detail in the Options documentation.

A.1 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.

A.2 Complex Types

This topic has these sections:

A.2.1 All Export Products

This section discusses information applicable to all products.

A.2.1.1 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>
```

A.2.1.2 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>
```

A.2.1.3 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>
```

A.2.1.4 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>
```

A.2.2 HTML Export

The following information is pertinent for HTML Export only.

A.2.2.1 AltLink

Valid for the altlink option. The 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>
```

A.2.2.2 DefaultFont

Valid for the defaultFont option. The 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>
```

A.2.2.3 FontFlags

Valid for the fontFlags option. It takes the form a 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>
```

A.2.3 Search Export

The following information is pertinent for Search Export only.

A.2.3.1 CharacterAttributes

This data type has been deprecated. The flags contained in it are now standalone Boolean options.

A.2.3.2 ParagraphAttributes

This 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>
```

A.2.3.3 SearchMLFlags

This data type has been deprecated. The flags contained in it are now standalone xsd:boolean options.

A.2.4 Image Export

The following information is valid for Image Export only.

A.2.4.1 DefaultFont

Valid for the defaultFont option. The 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>
```

A.2.4.2 DefaultMargins

Valid for the defaultMargins option. The 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>
```

A.2.4.3 TiffOptions

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>
```

A.3 Enumerations

This section covers these topics:

- Section A.3.1, "All Export Products"
- Section A.3.2, "HTML Export"
- Section A.3.3, "Search Export"
- Section A.3.4, "Image Export"
- Section A.3.5, "PDF Export"
- Section A.3.6, "XML Export"

A.3.1 All Export Products

This information is applicable for all products.

A.3.1.1 DefaultInputCharSetEnum

Valid for the defaultInputCharset option. 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"/>
```

A.3.1.2 DocumentMemoryModeEnum

Valid for the documentMemoryMode option. 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>
```

A.3.1.3 FallbackFormatEnum

Valid for the fallbackFormat option. The enumeration is defined as follows:

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

A.3.2 HTML Export

The following information is valid for HTML Export only.

A.3.2.1 CharacterByteOrderEnum

Valid for the characterByteOrder option. 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>
```

A.3.2.2 CharacterSetEnum

Valid for the outputCharacterSet option. 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>
```

A.3.2.3 ComplianceEnum

Valid for the compliance option. The enumeration is defined as follows:

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

A.3.2.4 EmailHeaderOutputEnum

EmailHeaderOutputEnum takes the place of the MimeHeaderOutputEnum. Valid for the emailHeaderOutput option. The enumeration is defined as follows:

```
<simpleType name="EmailHeaderOutputEnum">
  <restriction base="xsd:string">
```

```
    <enumeration value="emailHeaderStandard"/>
    <enumeration value="emailHeaderAll"/>
    <enumeration value="emailHeaderNone"/>
  </restriction>
</simpleType>
```

A.3.2.5 ExtractEmbeddedFilesEnum

Valid for the extractEmbeddedFiles option. The enumeration is defined as follows:

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

A.3.2.6 FlavorEnum

Valid for the flavor option. 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>
```

A.3.2.7 GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

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

A.3.2.8 GraphicTypeEnum

Valid for the graphicType option. 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>
```

A.3.2.9 GridAdvanceEnum

Valid for the gridAdvance option. The enumeration is defined as follows:

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

A.3.2.10 ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

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

A.3.2.11 SpreadSheetBordersEnum

Valid for the spreadsheetBorders option. The enumeration is defined as follows:

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

A.3.3 Search Export

The following information is valid for Search Export.

A.3.3.1 oleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

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

```
 </restriction>
</simpleType>
```

A.3.3.2 SearchMLUnmappedTextEnum

Valid for the unmappedText option. The enumeration is defined as follows:

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

A.3.3.3 XmlDefinitionMethodEnum

Valid for the xmlDefinitionMethod option. The enumeration is defined as follows:

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

A.3.4 Image Export

The following information is valid for Image Export.

A.3.4.1 DatabaseFitToPageEnum

Valid for the databaseFitToPage option. 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>
```

A.3.4.2 EmailHeaderOutputEnum

EmailHeaderOutputEnum takes the place of the MimeHeaderOutputEnum. Valid for the emailHeaderOutput option. The enumeration is defined as follows:

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

A.3.4.3 GraphicCroppingEnum

Valid for the graphicCropping option. The enumeration is defined as follows:

```
<simpleType name="GraphicCroppingEnum">
   <restriction base="string">
```

```
   <enumeration value="noCropping"/>
   <enumeration value="cropToContent"/>
</restriction>
```

A.3.4.4 GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

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

A.3.4.5 GraphicWatermarkScaleTypeEnum

Valid for the graphicWatermarkScaleType option. The enumeration is defined as follows:

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

A.3.4.6 MimeHeaderOutputEnum

Valid for the mimeHeaderOutput option. The mimeHeaderOutput option is no longer preferred, and has been replaced with the emailHeaderOutput option. The enumeration is defined as follows:

```
  <simpleType name="MimeHeaderOutputEnum">
    <restriction base="string">
      <enumeration value="all"/>
      <enumeration value="standard"/>
    <restriction>
  <simpleType>
```

A.3.4.7 ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

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

A.3.4.8 SpreadsheetFitToPageEnum

Valid for the spreadsheetFitToPage option. 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>
```

A.3.4.9 SpreadsheetPageDirectionEnum

Valid for the spreadsheetPageDirection option. The enumeration is defined as follows:

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

A.3.4.10 TiffByteOrderEnum

Part of the TiffOptions structure. The enumeration is defined as follows:

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

A.3.4.11 TiffColorSpaceEnum

Part of the TiffOptions structure. 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>
```

A.3.4.12 TiffCompressionEnum

Part of the TiffOptions structure. 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>
```

A.3.4.13 TiffFillOrderEnum

Part of the TiffOptions structure. The enumeration is defined as follows:

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

```
</simpleType>
```

A.3.5 PDF Export

The following information is valid for PDF Export.

A.3.5.1 DefaultPageUnitsEnum

Valid for the defaultPageUnits option. 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>
```

A.3.5.2 EmailHeaderOutputEnum

EmailHeaderOutputEnum takes the place of the MimeHeaderOutputEnum. Valid for the emailHeaderOutput option. The enumeration is defined as follows:

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

A.3.5.3 ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

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

A.3.5.4 WatermarkPositionEnum

Valid for the watermarkPosition option. The enumeration is defined as follows:

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

A.3.5.5 WatermarkScalingEnum

Valid for the watermarkScaling option. The enumeration is defined as follows:

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

```
 </restriction>
</simpleType>
```

A.3.6 XML Export

The following information is valid for XML Export.

A.3.6.1 GraphicSizeMethodEnum

Valid for the graphicSizeMethod option. The enumeration is defined as follows:

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

A.3.6.2 GraphicTypeEnum

Valid for the graphicType option. 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>
```

A.3.6.3 oleEmbeddingsEnum

Valid for the oleEmbeddings option. The enumeration is defined as follows:

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

A.3.6.4 ReorderMethodEnum

Valid for the reorderMethod option. The enumeration is defined as follows:

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

B

C/C++ Client Data Types

All options discussed in this chapter are described in detail in the Options documentation.

B.1 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

B.2 Complex Types

Complex types are listed by product:

B.2.1 All Export Products

This section discusses information that is pertinent for all products.

B.2.1.1 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

B.2.1.2 TS_char*

This data type is a pointer to the standard C data type char*.

B.2.1.3 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)

B.2.1.4 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;
```

B.2.1.5 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;
```

B.2.1.6 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

B.2.1.7 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;
```

B.2.2 HTML Export

The information in this section is pertinent for HTML Export only.

B.2.2.1 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;
```

B.2.2.2 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;
```

B.2.2.3 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;
```

B.2.3 Search Export

This information is valid for Search Export only.

B.2.3.1 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;
```

B.2.3.2 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;
```

B.2.3.3 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;
```

B.2.4 Image Export

This information is valid for Image Export only.

B.2.4.1 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;
```

B.2.4.2 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;
```

B.2.4.3 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;
```

B.3 Enumerations

Complex types are listed by product:

- Section B.3.1, "All Export Products"
- Section B.3.2, "HTML Export"
- Section B.3.3, "Search Export"
- Section B.3.4, "Image Export"
- Section B.3.5, "PDF Export"
- Section B.3.6, "XML Export"

B.3.1 All Export Products

This information is valid for all products.

B.3.1.1 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;
```

B.3.1.2 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;
```

B.3.1.3 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;
```

B.3.2 HTML Export

This information is valid for HTML Export only.

B.3.2.1 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;
```

B.3.2.2 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;
```

B.3.2.3 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;
```

B.3.2.4 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;
```

B.3.2.5 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;
```

B.3.2.6 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;
```

B.3.2.7 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;
```

B.3.2.8 OIT_GridAdvanceEnum

Valid for the gridAdvance option. The enumeration is defined as follows:

```
typedef enum OIT_GridAdvanceEnum
{
   oit_advanceAcross,
   oit_advanceDown
} OIT_GridAdvanceEnum;
```

B.3.2.9 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;
```

B.3.2.10 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;
```

B.3.2.11 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
};
```

B.3.3 Search Export

This information is valid for Search Export only.

B.3.3.1 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;
```

B.3.3.2 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
```

B.3.3.3 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;
```

B.3.4 Image Export

This information is valid for Image Export only.

B.3.4.1 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;
```

B.3.4.2 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;
```

B.3.4.3 OIT_GraphicCroppingEnum

Valid for the graphicCropping option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicCroppingEnum
{
   oit_noCropping,
   oit_cropToContent
```

```
} OIT_GraphicCroppingEnum;
```

B.3.4.4 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;
```

B.3.4.5 OIT_GraphicWatermarkScaleTypeEnum

Valid for the graphicWatermarkScaleType option. The enumeration is defined as follows:

```
typedef enum OIT_GraphicWatermarkScaleTypeEnum
{
  oit_scaleWatermarkOff,
  oit_scaleWatermarkByPercent
} OIT_GraphicWatermarkScaleTypeEnum;
```

B.3.4.6 OIT_MimeHeaderOutputEnum

Valid for the mimeHeaderOutput option. The enumeration is defined as follows:

```
typedef enum OIT_MimeHeaderOutputEnum
{
   oit_all,
   oit_standard,
} OIT_MimeHeaderOutputEnum
```

B.3.4.7 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;
```

B.3.4.8 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;
```

B.3.4.9 OIT_SpreadsheetPageDirectionEnum

Valid for the spreadsheetPageDirection option. The enumeration is defined as follows:

```
typedef enum OIT_SpreadsheetPageDirectionEnum
```

```
{
   oit_downThenAcross,
   oit_acrossThenDown
} OIT_SpreadsheetPageDirectionEnum;
```

B.3.4.10 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;
```

B.3.4.11 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;
```

B.3.4.12 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;
```

B.3.4.13 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;
```

B.3.5 PDF Export

This information is valid for PDF Export only.

B.3.5.1 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;
```

B.3.5.2 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;
```

B.3.5.3 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;
```

B.3.5.4 OIT_WatermarkPositionEnum

Valid for the watermarkPosition option. The enumeration is defined as follows:

```
typedef enum OIT_WatermarkPostionEnum
{
   oit_centerOfPage
} OIT_WatermarkPostionEnum;
```

B.3.5.5 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;
```

B.3.6 XML Export

This information is valid for XML Export only.

B.3.6.1 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;
```

B.3.6.2 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;
```

B.3.6.3 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;
```

B.3.6.4 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;
```

B.3.6.5 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.

C.1 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

C.2 Complex Types

This topic is listed by product:

C.2.1 All Products

This information pertains to all products.

C.2.1.1 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)

C.2.1.2 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 Section C.3.2.2, "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

C.2.1.3 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;
}
```

C.2.2 HTML Export

The following information pertains to HTML Export.

C.2.2.1 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;
}
```

C.2.2.2 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;
};
```

C.2.2.3 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;
}
```

C.2.3 Search Export

The following information applies to Search Export.

C.2.3.1 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;
```

```
}
```

C.2.3.2 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;
}
```

C.2.3.3 SearchMLFlags

This data type has been deprecated. The flags contained in it are now standalone Boolean options.

C.2.4 Image Export

This information applies to Image Export.

C.2.4.1 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;
};
```

C.2.4.2 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;
}
```

C.2.4.3 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;
}
```

C.3 Enumerations

This topic has these sections:

C.3.1 All Export

This information applies to all products.

C.3.1.1 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");
```

C.3.1.2 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");
```

C.3.1.3 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");
```

C.3.2 HTML Export

This information applies to HTML Export.

C.3.2.1 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");
```

C.3.2.2 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");
```

C.3.2.3 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");
```

C.3.2.4 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");
```

C.3.2.5 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");
```

C.3.2.6 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");
```

C.3.2.7 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");
```

C.3.2.8 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");
```

C.3.2.9 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");
```

C.3.2.10 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")
```

C.3.3 Search Export

This information applies to Search Export.

C.3.3.1 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");
```

C.3.3.2 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");
```

C.3.3.3 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");
```

C.3.4 Image Export

This information applies to Image Export.

C.3.4.1 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");
```

C.3.4.2 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");
```

C.3.4.3 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");
```

C.3.4.4 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");
```

C.3.4.5 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")
```

C.3.4.6 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");
```

C.3.4.7 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");
```

C.3.4.8 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");
```

C.3.4.9 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");
```

C.3.4.10 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");
```

C.3.4.11 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");
```

C.3.4.12 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");
```

C.3.5 PDF Export

This information applies to PDF Export.

C.3.5.1 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;
```

C.3.5.2 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");
```

C.3.5.3 WatermarkPositionEnum

Valid for the watermarkPosition option. The class WatermarkPositionEnum defines the following static members:

```
public static final WatermarkPositionEnum CENTEROFPAGE = new
WatermarkPositionEnum("centerOfPage");
```

C.3.5.4 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");
```

C.3.6 XML Export

This information applies to XML Export.

C.3.6.1 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");
```

C.3.6.2 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");
```

C.3.6.3 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");
```

C.3.6.4 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");
```

C.3.6.5 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.

D.1 Outside In Transformation Server Licensing

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

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 Wegner

Portions 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 kXML2

Copyright (c) 2002,2003, Stefan Haustein, Oberhausen, Rhld., Germany

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

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

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR 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 archives

License for use and distribution of FREE portable version

The 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

J

L

M

O

P

R

S

T

U

V

W

X