Oracle® Outside In Content Access
Developer's Guide

Release 8.5.4
E80945-01
May 2018

Oracle Outside In Content Access Developer's Guide, Release 8.5.4

E80945-01

Primary Author: Nirmala Suryaprakasha

Contents

7 Content Description

8 Redirected IO

9 Implementation Issues

10 Sample Applications

A Content Access Options

Index

Preface

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

Audience

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

Related Documents

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

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

Content Access is part of Oracle's family of OEM products known as Outside In Technology, a powerful document extraction, conversion and viewing technology that can access the information in more than 600 file formats. Content Access is a server-grade technology that provides developers with normalized access to content stored in documents across multiple platforms.

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

> **Note:**
>
> For new functionality information, see What's New guide.

This chapter includes the following sections:

1.1 What Does This Technology Do?

Outside In Content Access provides a simple interface to extract text and metadata from business documents. This technology is particularly useful for document indexing applications. The product is comprised of two modules: Content Access and Text Access. Benefits include:

- The ability to extract text from documents with automatic translation into a particular character set, such as Unicode or ANSI.
- Access to numerous additional properties of documents that store information such as author, keywords, typist, version notes, carbon copy, checked by, subject, character and paragraph attributes, and so forth.
- A common interface to the content of diverse file formats including word processing, spreadsheet, database, email, vector, and presentation formats.
- The Text Access module's specific functions have tight integration with Outside In Technology, such that text generated by the text access functions is highlighted in the Viewer.

- Text Access and Content Access generate the same raw text. However, the following points are important.
  - rawtext and Text Access will extract some text as unmappable characters because they cannot be annotated. This includes text that is not visible (for example, document properties, hidden text, and so on.).
  - rawtext and Text Access only operate on the top-most layer of the file, and will not extract text from embedded documents. Thus, not all visible text will be extractable via rawtext or Text Access.
  - Content Access can be used to extract hidden text, like document properties; and text from embedded documents.
  - It should be noted that other Outside In products offer powerful text extraction and tagging abilities, such as Search Export and XML Export.

1.2 Architectural Overview

The basic architecture of Content Access is the same across all supported platforms:

Filter/Module	Description
Input Filter	The input filters form the base of the architecture. Each one reads a specific file format or set of related formats and sends the data to the chunker module through a standard set of function calls. There are more than 150 of these filters that read more than 600 distinct file formats. Filters are loaded on demand by the data access module.
Chunker	The Chunker module is responsible for caching a certain amount of data from the filter and returning this data to the Content Access module.
Content Access	The Content Access module reads data from the chunker and repackages it in a way that is convenient for the developer. This repackaging process includes mapping characters to a particular character set and converting some data (such as paragraph and cell breaks) into representative characters. CA outputs non-visible text, provides a wealth of style information, provides the information needed for the consumer to process sub-documents, and optionally produces non-textual information such as numbers in spreadsheets.
Text Access	The Text Access module is similar to the Content Access module, although it is restricted to text. For more information, see Text Access Functions.
Data Access	The Data Access module implements a generic API for access to files. It understands how to identify and load the correct filter for all the supported file formats. The module delivers to the developer a generic handle to the requested file, which can then be used to run more specialized processes. The Data Access module is responsible for providing a document for the Content Access module. Data Access conserves resources by creating only one file handle and one chunker handle for each file, even if it is opened in multiple Content Access instances. It also provides a unified platform for several modules in addition to Content Access, including Text Access and Remote Filter Access.

1.3 Definition of Terms

The following table provides definitions of some common terms.

Term	Definition
Developer	Someone integrating this technology into another technology or application. Most likely this is you, the reader.
Source File	The file the developer wishes to extract content from.
Data Access Module	The core of Outside In Data Access, in the SCCDA library.
Data Access Submodule (also referred to as "Submodule")	This refers to any of the Outside In Data Access modules, including SCCCA (Content Access) and SCCTA (Text Access), but excluding SCCDA (Data Access).
Document Handle (also referred to as "`hDoc`")	A Document Handle is created when a file is opened using Data Access (see Data Access Common Functions). Each Document Handle may have any number of Subhandles.
Content Handle (also referred to as "`hItem`")	The handle created by a call to CAOpenContent or TAOpenText. Every Content Handle has a Document Handle associated with it. The DASetOption and DAGetOption functions in the Data Access Module may be called with any Content Handle or Document Handle. The DARetrieveDocHandle function returns the Document Handle associated with any Content Handle.

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, ca/sdk/ca_win-x86-32_sdk). Under each of these directories are the following two subdirectories:

- **redist**: Contains only the files that the customer is allowed to redistribute. These include all the compiled modules, filter support files, .xsd and .dtd files, cmmap000.bin, and third-party libraries, like freetype.
- **sdk**: Contains the other subdirectories that used to be at the root-level of an sdk (common, lib (windows only), resource, samplefiles, and samplecode (previously samples). In addition, one new subdirectory has been added, demo, that holds all of the compiled sample apps and other files that are needed to demo the products. These are files that the customer should not redistribute (.cfg files, exportmaps, and so forth).

In the root platform directory (for example, ca/sdk/ca_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 How to Use Content Access

Here's a step-by-step overview of how to obtain information from a source file using Content Access.

1. Call DAInitEx to initialize the Data Access technology. This function needs to be called only once per application. If using threading, then pass in the correct ThreadOption.
2. Set "Null" options: Certain options need to be set before the desired source file is opened. These options are identified by requiring a NULL handle type. They include, but aren't limited to:
   - SCCOPT_FALLBACKFORMAT
   - SCCOPT_FIFLAGS
   - SCCOPT_TEMPDIR
3. Open the Source File: DAOpenDocument is called to create a document handle that uniquely identifies the source file. This handle may be used in subsequent calls to the CAOpenContent function or the open function of any other Data Access Submodule, and will be used to close the file when access is complete. This allows the file to be accessed from multiple Data Access Submodules without reopening.
4. Set other Options: Once the source document has been opened, set any other desired options. Most options will be set at this time and are identified by requiring a VTHDOC handle type.
5. Open a Handle to Content Access: Using the document handle, CAOpenContent is called to obtain a content handle that identifies the file to the Content Access module. This handle will used in all subsequent calls to the Content Access functions.
6. Retrieve the first Information from the File: Call CAReadFirst to read the first piece of information from the file. Note: this step may be repeated to reread the file.
7. Retrieve other Information from the File: Repeatedly call CAReadNext, which will iteratively read through and process the file.
8. Process sub-documents (Optional): When you encounter a sub-document, you may process that sub-document by repeating steps 4-10. Sub-documents are identified by either the SCCCA_OBJECT type or the SCCCA_LINKEDOBJECT subtype of the SCCCA_BEGINTAG type. Note: the document handle and content handle will be different for the parent and sub-document.
9. Close the Content Access Handle: Call CACloseContent to terminate the content access for the file. After this function is called, the content handle will no longer be valid, but the document handle may still be used.
10. Close the Source File: DACloseDocument is called to close the source file. After calling this function, the document handle will no longer be valid.
11. De-initialize DA: DADeInit is called to de-initialize the Data Access technology.

1.6 How to Use Text Access

Here's a step-by-step overview of how to obtain information from a source file using Text Access.

1. Call DAInitEx to initialize the Data Access technology. This function needs to be called only once per application. If using threading, then pass in the correct ThreadOption.
2. Set "Null" options: Certain options need to be set before the desired source file is opened. These options are identified by requiring a NULL handle type. They include, but aren't limited to:
   - SCCOPT_FALLBACKFORMAT
   - SCCOPT_FIFLAGS
   - SCCOPT_TEMPDIR
3. Open the Source File: DAOpenDocument is called to create a document handle that uniquely identifies the source file. This handle may be used in subsequent calls to the TAOpenText function or the open function of any other Data Access Submodule, and will be used to close the file when access is complete. This allows the file to be accessed from multiple Data Access Submodules without reopening.
4. Set other Options: Once the source document has been opened, set any other desired options. Most options will be set at this time and are identified by requiring a VTHDOC handle type.
5. Open a Handle to Text Access: Using the document handle, TAOpenContent is called to obtain a content handle that identifies the file to the Text Access module. This handle will used in all subsequent calls to the Text Access functions.
6. Retrieve the first Information from the File: Call TAReadFirst to read the first piece of information from the file. Note: this step may be repeated to reread the file.
7. Retrieve other Information from the File: Repeatedly call TAReadNext, which will iteratively read through and process the file.
8. Close the Text Access Handle: Call TACloseText to terminate the text access for the file. After this function is called, the text handle will no longer be valid, but the document handle may still be used.
9. Close the Source File: DACloseDocument is called to close the source file. After calling this function, the document handle will no longer be valid.
10. De-initialize DA: DADeInit is called to de-initialize the Data Access technology.

2
Windows Implementation Details

This chapter describes the implementation of the Content Access SDK on the Windows platform. Content Access is delivered as a set of DLLs.

For a list of the currently supported platforms, see Outside In Technology and click links under Certified Platforms and Supported Formats from the Get Started page.

This chapter includes the following sections:

- See Installation for information.
- Libraries and Structure
- The Basics
- Character Sets
- Runtime Considerations
- Changing Resources

2.1 Libraries and Structure

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

API DLLs

These DLLs implement the API. They should be linked with the developer's application. LIB files are included in the SDK.

File	Description
sccca.dll	Content Access module (provides organized chunker data for the developer)
sccda.dll	Data Access module
sccfi.dll	File Identification module (identifies files based on their contents). The File ID Specification may not be used directly by any application or workflow without it being separately licensed expressly for that purpose.
sccta.dll	Text Access module (provides straight text data for the developer)

Support DLLs

File	Description
sccch.dll	Chunker (provides caching of and access to filter data for the display engine)
sccfa.dll	Filter Access module

File	Description
sccfmt.dll	Formatting module (resolves numbers to formatted strings)
sccfut.dll	Filter utility module
sccind.dll	Indexing engine
scclo.dll	Localization library (all strings, menus, dialogs and dialog procedures reside here)
sccole.dll	OLE rendering module
sccut.dll	Utility functions (including IO subsystem)
wvcore.dll	The GDI Abstraction layer

Filter DLLs

File	Description
vs*.dll	Filters for specific file types (there are more than 150 of these filters, covering more than 600 file formats)
oitnsf.id	Support file for the vsnsf filter.

Premier Graphics Filters

File	Description
i*2.dll	Import filters for premier graphics formats
isgdi32.dll	Interface to premier graphics filters

Additional Files

File	Description
adinit.dat	Support file for the vsacad filter
cmmap000.bin	Tables for character mapping (all character sets)
cmmap000.sbc	Tables for character mapping (single-byte character sets). Located in the common directory.
cmmap000.dbc	Identical to cmmap000.Bin, but renamed for clarity (.dbc = double-byte character). This file is located in the common directory.
compreg.bin	Outside In Component Registry

2.2 The Basics

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

For detailed information about all sample applications included with this product, see Sample Applications.

2.2.1 What You Need in Your Source Code

Any source code that uses this product should `#include` the file sccca.h (for Content Access) and/or sccta.h (for Text Access) and `#define WINDOWS` and `WIN32` or `WIN64`. For example, a Windows application might have a source file with the following lines:

```
#define WINDOWS         /* Will be automatically defined if your
                           compiler defines _WINDOWS */
#define WIN32
#include <sccca.h>      /* If using ContentAccess */
#include <sccta.h>      /* If using Text Access */
```

The developer's application should be linked to the Content Access (and/or Text Access) and Data Access DLLs through the provided libraries (sccta.lib, sccca.lib and sccda.lib).

2.2.2 Options and Information Storage

One set of information is created by the technology, the default options. In the Windows implementation, this is built by the technology as needed, usually the first time the product is run. You do not need to ship this list with your application. The list is automatically regenerated if corrupted or deleted.

The files used to store this information are stored in a .oit subdirectory in the following location:

\Documents and Settings*user name*\Application Data

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

The file is:

*.d = Display engine lists

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

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

2.2.3 Structure Alignment

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

2.3 Character Sets

This section provides information about character sets.

2.3.1 Default API Character Set

The strings passed in the Windows API are ANSI1252 by default.

2.3.2 Double-Byte Character Set Mapping

Please note that to optimize performance on systems that do not require DBCS support, a second character mapping bin file, that does not contain any of the DBCS pages, is now included. The second bin file will give additional performance benefits for English documents, but will not be able to handle DBCS documents. To use the new bin file, replace the cmmap000.bin with the new bin file, cmmap000.sbc. For clarity, a copy of the cmmap000.bin file named cmmap000.dbc has also been included. Both the cmmap000.sbc and cmmap000.dbc files are located in the \sdk \common directory of the technology.

2.4 Runtime Considerations

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

2.5 Changing Resources

Outside In Content Access ships with the necessary files for OEMs to change any of the strings in the technology as they see fit.

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

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

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

1. Compile the .res file:

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

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

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

> **Note:**
>
> Developers should make sure they have set up their environment variables to build the library for their specific architecture. For Windows x86_32, when compiling with VS 2013, the solution is to run vsvars32.bat (in a standard VS 2013 installation, this is found in C:\Program Files\Microsoft Visual Studio X\Common7\Tools\). If this works correctly, you will see the statement, "Setting environment for using Microsoft Visual Studio 2013 tools." If you do not complete this step, you may have conflicts that lead to unresolved symbols due to conflicts with the Microsoft CRT.

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

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

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

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

3
UNIX Implementation Details

This chapter describes the UNIX implementation of the Content Access SDK on the UNIX platform. The UNIX implementation of Content Access is delivered as a set of shared libraries.
For a list of the currently supported platforms, see Outside In Technology and click links under Certified Platforms and Supported Formats from the Get Started page.

This chapter includes the following sections:

3.1 Installation

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

```
gunzip tgzfile
tar xvf tarfile
```

The installation directory should contain the following directory structure:

Directory	Description
/redist	Contains a working copy of the UNIX version of the technology.
/sdk/common	Contains the C include files needed to build or rebuild the technology.
/sdk/demo	Contains the compiled executables of the sample applications.
/sdk/resource	Contains localization resource files. For more information, see Changing Resources.
/sdk/samplecode	Contains a subdirectory holding the source code for a sample application. For more information, see Sample Applications.

Directory	Description
/sdk/samplefiles	Contains sample files designed to exercise the technology.

3.1.1 NSF Support

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

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

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

  [Notes]
- Add the Lotus Domino directory to the `$LD_LIBRARY_PATH` environment variable.
- Set the environment variable `$Notes_ExecDirectory` to the Lotus Domino directory.

3.2 Libraries and Structure

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

The following is a brief description of the included libraries and support files. Note that in instances where a file extension is listed as .*, the file extension will vary for each UNIX platform (**sl** on HP/UX, **so** on Linux and Solaris).

API Libraries

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

File	Description
libsc_ca.*	Content Access module (provides organized chunker data for the developer)
libsc_da.*	Data Access module
libsc_fi.*	File Identification module (identifies files based on their contents). **The File ID Specification may not be used directly by any application or workflow without it being separately licensed expressly for that purpose.**

File	Description
libsc_ta.*	Text Access module (provides straight text data for the developer)

Support Libraries

File	Description
libsc_ch.*	Chunker (provides caching of and access to filter data for the display engine)
libsc_fa.*	Filter Access module
libsc_fmt.*	Formatting module (resolves numbers to formatted strings)
libsc_fut.*	Filter utility module
libsc_ind.*	Indexing engine
libsc_lo.*	Localization library (all strings, menus, dialogs and dialog procedures reside here)
libsc_ut.*	Utility functions, including IO subsystem
libsc_xp.*	XPrinter bridge
libwv_core.*	The Abstraction layer

Filter Libraries

File	Description
libvs_*.*	Filters for specific file types (there are more than 150 of these filters, covering more than 600 file formats)

Premier Graphics Filters

File	Description
libi*.*	These 30 files are the import filters for premier graphics formats.
libis_unx2.*	Interface to premier graphics filters

Additional Files

File	Description
adinit.dat	Support file for the vsacad and vsacd2 filters
cmmap000.bin	Tables for character mapping (all character sets)
cmmap000.sbc	Tables for character mapping (single-byte character sets). This file is located in the common directory.
cmmap000.dbc	Identical to cmmap000.Bin, but renamed for clarity (.dbc = double-byte character). This file is located in the *common* directory.
oitnsf.id	Support file for the vsnsf filter.

3.3 The Basics

All the steps outlined in this section are used in the sample applications provided with the SDK. Looking at the code for the **casample** sample application (see Sample Applications) is recommended for a real world example of this process.

3.3.1 What You Need in Your Source Code

Any source code that uses this product should `#include` the file sccca.h (for Content Access) and/or sccta.h (for Text Access) and `#define UNIX`. For example, a 32-bit UNIX application might have a source file with the following lines:

```
#define UNIX
#include <sccca.h>      /* If using ContentAccess */
#include <sccta.h>      /* If using Text Access */
```

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

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

3.3.2 Options and Information Storage

Three sets of information are created by the technology: the default options, a list of available filters and a list of available display engines. In the UNIX implementations, these lists are built as needed, usually the first time the product is run. You do not need to ship these lists with your application.

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

The files are:

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

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

3.4 Character Sets

This section provides information about character sets.

3.4.1 Default API Character Set

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

3.4.2 Double-Byte Character Set Mapping

To optimize performance on systems that do not require DBCS support, a second character mapping bin file not containing any of the DBCS pages is now included. The second bin file gives additional performance benefits for English documents, but will not be able to handle DBCS documents. To use the new bin file, replace the cmmap000.bin with the new bin file, cmmap000.sbc. For clarity, a copy of the cmmap000.bin file named cmmap000.dbc has also been included. Both the cmmap000.sbc and cmmap000.dbc files are located in the /common directory of the technology.

3.5 Runtime Considerations

This section provides information about runtime considerations.

3.5.1 Signal Handling

This product traps and handles the following signals:

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

To override the default handling of these signals you can set your own signal handlers. This can be done after the developer's application has called DAInitEx().

> **Note:**
>
> The Java Native Interface (JNI) allows Java code to call and be called by native code (C/C++ in the case of OIT). You may run into problems if Java isn't allowed to handle signals and forward them to OIT. If OIT catches the signals and forwards them to Java, the JVMs will sometimes crash. OIT installs signal handlers when DAInitEx() is called, so if you call OIT after the JVM is created, you will need to use libjsig. Refer here for more information:
>
> `http://www.oracle.com/technetwork/java/javase/index-137495.html`

3.5.2 Runtime Search Path and $ORIGIN

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

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

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

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

3.6 Environment Variables

There are a number of environment variables the UNIX implementation of the technology may use at run time. While described elsewhere, following is a short summary of those variables and their usage.

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

3.7 Changing Resources

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

Along with lodlgstr.h, an object file, scclo.o has been provided that is necessary for the linking phase of the build. A makefile has also been provided for building the library. The makefile allows building on all of the UNIX platforms supported by Outside In. It may be necessary to make minor modifications to the makefile so that the system header files and libraries can be found for compiling and linking. There are standard INCLUDE and LIB `make` variables defined for each platform in the makefile. Edit these variables to point to the header files and libraries on your particular system. Other make variables are:

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

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

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

All dialog boxes are created directly in the viewer code internally and are compiled and linked in the normal compilation process. There are no separate resource files corresponding to the .rc files in the Windows code.

Additional viewer resources are defined in xsccvw.h, which is included in the code for the sample executables and the viewer.

3.8 HP-UX Compiling and Linking

In the following example, libsc_ca.sl and libsc_da.sl are the only libraries that need to be linked with the casample. Not all applications that use the Content Access module will require the use of these libraries. They can be loaded when the application starts by linking them directly at compile time or they can be loaded dynamically by your application using library load functions (for example, shl_load).

The following are example command lines used to compile the sample application casample from the /sdk/samplecode/unix directory. The command lines are separated into sections for HP/UX and HP/UX on Itanium (which requires GCC). Please note that this command line is only an example. The actual command line required on the developer's system may vary. The example assumes that the include and library file search paths for the technology libraries and any required X libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L *library file path* options, respectively, so that the compiler and linker can locate all required files.

HP-UX on RISC

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

HP-UX on Itanium (64 bit)

```
cc -w -o ../casample/unix/casample ../casample/unix/casample.c +DD64 -I../../common -
L../../demo -L/usr/lib/hpux64 -lsc_da -lsc_ca -DUNIX -DUNIX_64 -Wl,+s,+b,'$ORIGIN'
```

3.9 IBM AIX Compiling and Linking

All libraries should be installed into a single directory and the directory must be included in the system's shared library path ($LIBPATH) as well as the executable path ($PATH).

> **Note:**
>
> $LIBPATH must be set and must point to the directory containing the Outside In technology.

Outside In Technology has been updated to increase performance, at a cost of using more memory. It is possible that this increased memory usage may cause a problem on AIX systems, which can be very conservative in the amount of memory they grant to processes. If your application experiences problems due to memory limitations with Outside In, you may be able to fix this problem by using the "large page" memory model. If you anticipate viewing or converting very large files with Outside In technology, we recommend linking your applications with the -bmaxdata flag (for example, ‘cc -o foo foo.c -bmaxdata:0x80000000'). If you are currently seeing illegal instruction errors followed by immediate program exit, this is probably due to not using the large data model.

The following is an example command line used to compile the sample application casample from the /sdk/samplecode/unix directory. This command line is only an example. The actual command line required on the developer's system may vary. The example assumes that the include and library file search paths for the technology libraries and any required X libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L *library file path* options, respectively, so that the compiler and linker can locate all required files. Developers need to pass -brtl to the linker to list libraries in the link command as dependencies of their applications.

> **Note:**
>
> Developers may need to use the -qcpluscmt flag to allow C++ style comments.

```
gcc -w -o ../casample/unix/casample ../casample/unix/casample.c -I../../common -
L../../demo -lsc_ca -lsc_da -DUNIX -DFUNCPROTO -Wl,-brtl
```

3.10 Linux Compiling and Linking

This section provides information about Linux compiling and linking.

3.10.1 Library Compatibility

This section provides information about library compatibility.

3.10.1.1 Motif Libraries

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.3, needed by ../../libsc_vw.so, may conflict with libXm.so.2
```

The proper solution to this problem 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://rpmfind.net/linux/RPM.

If you are doing development, you will also need the proper header files, as well.

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

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

> **Note:**
>
> If a directory needs to be specified for the compiler to find the shared libraries, it is recommended that the $LD_LIBRARY_PATH environment variable be used. This will prevent the compiler from hard-coding the library's current directory into the executable as the only directory to search for the library at run time. Instead, the system will first search the directories specified by $LD_LIBRARY_PATH for the library.

3.10.1.2 GLIBC and Compiler Versions

For each Linux platform supported by Outside In, the following table indicates the compiler version used and the minimum required version of the GNU standard C library upon which Outside In depends.

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

3.10.1.3 Other Libraries

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

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

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

3.10.2 Compiling and Linking

In the following example, the libsc_ca.so and libsc_da.so are the only libraries needing to be linked with the casample. Not all applications that use the Content Access module will require the use of all of these libraries. They can be loaded when the application starts by linking them directly at compile time or they can be loaded dynamically by your application using library load functions (for example, dlopen).

The following is an example command line used to compile the sample application **casample** from the /sdk/samplecode/unix directory. Please note that this command line is only an example. The actual command line required on the developer's system may vary. The example assumes that the include and library file search paths for the technology libraries and any required X libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L *library file path* options, respectively, so the compiler and linker can locate all required files.

Linux 32-bit (includes Linux PPC)

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

Linux 64-bit

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

Linux zSeries

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

3.11 Oracle Solaris Compiling and Linking

All libraries should be installed into a single directory.

> **Note:**
>
> This product does not support the old Solaris BSD mode.

In the following example, the libsc_ca.so and libsc_da.so are the only libraries that need to be linked with the casample. Not all applications that use the Content Access module will require the use of all of these libraries. They can be loaded when the application starts by linking them directly at compile time or they can be loaded dynamically by your application using library load functions (for example, dlopen).

The following is an example command line used to compile the sample application casample from the /sdk/samplecode/unix directory. Please note that this command line is only an example. The actual command line required on the developer's system may vary. The example assumes that the include and library file search paths for the technology libraries and any required X libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L *library file path* options, respectively, so that the compiler and linker can locate all required files.

> **Note:**
>
> Developers may need to use the -xcc flag to allow C++ style comments.

3.11.1 Oracle Solaris SPARC

```
cc -I/usr/include -I/usr/dt/share/include -I../../common -w -o ../casample/unix/
casample ../casample/unix/casample.c  -L../../demo -L/usr/lib -L/lib -lc -ldl -
lsc_ca -lsc_da -DUNIX -Wl,-R,'$ORIGIN'
```

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

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

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

- For Solaris 9 - Patch 111722-04

3.11.2 Oracle Solaris x86

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

```
cc -I/usr/include -I/usr/dt/share/include -I../../common -w -o ../casample/unix/
casample ../casample/unix/casample.c -L../../demo -L/usr/lib  -lsc_ca -lsc_da -DUNIX
-R '$ORIGIN'+ -DUNIX
```

3.12 FreeBSD Compiling and Linking

The following is an example command line used to compile the sample application casample from the /sdk/samplecode/unix directory. Please note that this command line is only an example. The actual command line required on the developer's system may vary. The example assumes that the include and library file search paths for the technology libraries and any required X libraries are set correctly. If they are not set correctly, the search paths for the include and/or library files must be explicitly specified via the -I *include file path* and/or -L *library file path* options, respectively, so the compiler and linker can locate all required files.

```
gcc -w -o ../casample/unix/casample ../casample/unix/casample.c -I/usr/local/include
-I../../common -L../../demo -L/usr/local/lib -lsc_da -lsc_ca -DUNIX -Wl,-rpath,../../
demo
```

4

Data Access Common Functions

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

4.1 Deprecated Functions

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

4.2 DAInitEx

This function tells the Data Access module to perform any necessary initialization it needs to prepare for document access. This function must be called before the first time the application uses the module to retrieve data from any document. This function supersedes the old DAInit and DAThreadInit functions.

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

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

> **Note:**
>
> Multiple threads are supported for all Windows platforms, the 32-bit versions of Linux x86 and Solaris SPARC, Linux x64 and Solaris SPARC 64. Failed initialization of the threading function will not impair other API calls. If threading isn't initialized or fails, stub functions are called instead of mutex functions.

Prototype

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

Parameters

- ThreadOption: can be one of the following values:
  - DATHREAD_INIT_NOTHREADS: No thread support requested.
  - DATHREAD_INIT_PTHREADS: Support for PTHREADS requested.
  - DATHREAD_INIT_NATIVETHREADS: Support for native threading requested. Supported only on Microsoft Windows platforms and Oracle Solaris.

- dwFlags: can be one or more of the following flags OR-ed together
  - OI_INIT_DEFAULT: Options Load and Save are performed normally
  - OI_INIT_NOSAVEOPTIONS: The options file will not be saved on exit
  - OI_INIT_NOLOADOPTIONS: The options file will not be read during initialization.

Return Values

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

4.3 DADeInit

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

Prototype

```
DAERR DADeInit();
```

Return Values

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

4.4 DAOpenDocument

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

Prototype

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

Parameters

- lphDoc: Pointer to a handle that will be filled with a value that uniquely identifies the document to data access. The developer will use this handle in subsequent calls to data access to identify this particular source file.

  This is *not* an operating system file handle.
- dwSpecType: Describes the contents of pSpec. Together, dwSpecType and pSpec describe the location of the source file.

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

Must be one of the following values:

- IOTYPE_ANSIPATH: Windows only. pSpec points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) file name conventions.
- IOTYPE_UNICODEPATH: Windows only. pSpec points to a NULL-terminated full path name using the Unicode character set and NTFS (Win32 and Win64) file name conventions.
- IOTYPE_UNIXPATH: X Windows on UNIX platforms only. pSpec points to a NULL-terminated full path name using the system default character set and UNIX path conventions. Unicode paths can be accessed on UNIX platforms by using a UTF-8 encoded path with IOTYPE_UNIXPATH.
- IOTYPE_SUBOBJECT: All platforms. Opens an embedded object for data access. pSpec points to a structure IOSPECSUBOBJECT (see IOSPECSUBOBJECT Structure) that has been filled with values returned in a SCCCA_OBJECT content entry from Content Access.
- IOTYPE_REDIRECT: All platforms. pSpec points to a developer-defined struct that allows the developer to redirect the IO routines used to read the file.
- IOTYPE_ARCHIVEOBJECT: All platforms. Opens an embedded archive object for data access. pSpec points to a structure IOSPECARCHIVEOBJECT (see IOSPECARCHIVEOBJECT Structure) that has been filled with values returned in a SCCCA_OBJECT content entry from Content Access.
- IOTYPE_LINKEDOBJECT: All platforms. Opens an object specified by a linked object for data access. pSpec points to a structure IOSPECLINKEDOBJECT (see IOSPECLINKEDOBJECT Structure) that has been filled with values returned in an SCCCA_BEGINTAG or SCCCA_ENDTAG with a subtype of SCCCA_LINKEDOBJECT content entry from Content Access.
- IOTYPE_OBJECT: All platforms. Opens an object (archive, embedded, or linked) for data access. pSpec points to a structure SCCDAOBJECT (see SCCDAOBJECT Structure) that has been filled with values from Content Access (SCCCA_OBJECT or SCCCA_BEGINTAG with a subtype of SCCCA_LINKEDOBJECT) or from the <document> element in the SearchML flavor of Search Export.

- pSpec: File location specification.
- dwFlags: The low WORD is the file ID for the document (0 by default). If you set the file ID incorrectly, the technology will fail. If set to 0, the file identification technology will determine the input file type automatically. The high WORD should be set to 0. It may also be set to the following flags:

- DAOPENDOCUMENT_ARCHIVEONLYMODE: This flag may only be used with archive files. It opens the archive in a special mode that is only usable with DASaveRandomTreeRecord and DAOpenRandomTreeRecord.
- DAOPENDOCUMENT_CONTINUEONFAILURE: Some embeddings may have both an OLE representation and an alternate graphic. When this flag is set for `IOTYPE_OBJECT`, the technology will first try to access the OLE representation. If there are errors, it will then attempt to access the alternate graphic.

Return Values

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

4.4.1 IOSPECSUBOBJECT Structure

```
typedef struct IOSPECSUBOBJECTtag
      {
      VTDWORD      dwStructSize;
      VTSYSPARAM   hDoc;           /* Parent Doc hDoc */
      VTDWORD      dwObjectId;     /* Object Identifier */
      VTDWORD      dwStreamId;     /* Stream Identifier */
      VTDWORD      dwReserved1;    /* Must always be 0 */
      VTDWORD      dwReserved2;    /* Must always be 0 */
      } IOSPECSUBOBJECT, * PIOSPECSUBOBJECT;
```

4.4.2 IOSPECLINKEDOBJECT Structure

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

4.4.3 IOSPECARCHIVEOBJECT Structure

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

4.4.4 SCCDAOBJECT Structure

```
typedef struct SCCDAOBJECTtag
{
   VTDWORD   dwSize;         /* sizeof(SCCDAOBJECT) */
   VTHDOC    hDoc;           /* DA handle for the document
                                containing the object */
   VTDWORD   dwObjectType;   /* SCCCA_EMBEDDEDOBJECT,
                                SCCCA_LINKEDOBJECT,
                                SCCCA_COMPRESSEDFILE or
                                SCCCA_ATTACHMENT */
   VTDWORD   dwData1;        /* Data identifying the object */
   VTDWORD   dwData2;        /* Data identifying the object */
   VTDWORD   dwData3;        /* Data identifying the object */
   VTDWORD   dwData4;        /* Data identifying the object */
} SCCDAOBJECT, * PSCCDAOBJECT;
```

4.5 DACloseDocument

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

Prototype

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

Parameters

- hDoc: Identifier of open document. Must be a handle returned by the DAOpenDocument function.
- DAERR_OK: Returned if close succeeded. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.

4.6 DARetrieveDocHandle

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

Prototype

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

Parameters

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

Return Value

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

4.7 DASetOption

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

Prototype

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

Parameters

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

  If this parameter is NULL, then setting the option will affect all documents opened thereafter. Once an option is set using the NULL handle, this option becomes the default option thereafter. Note that this parameter should only be set to NULL if the option being set can take that value.
- dwOptionId: The identifier of the option to be set.
- pValue: Pointer to a buffer containing the value of the option.
- dwValueSize: The size in bytes of the data pointed to by pValue. For a string value, the NULL terminator should be included when calculating dwValueSize.

Return Value

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

4.8 DAGetOption

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

Prototype

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

Parameters

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

Return Value

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

4.9 DAGetFileId

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

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

Prototype

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

Parameters

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

Return Value

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

4.10 DAGetFileIdEx

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

Prototype

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

Parameters

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

Return Value

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

Values with RAWFI turned off

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

Values with RAWFI turned on

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

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

4.11 DAGetErrorString

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

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

Parameters

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

Return Value

- none

4.12 DAGetObjectInfo

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

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

Parameters

- hDoc: The handle returned by `DAOpenDocument`.
- dwInfoId

  The identifier of the requested information. Can be any of the following values:

  - DAOBJECT_NAME_A: Retrieves the name of the object, in 8-bit characters. pInfo points to a buffer of size DA_PATHSIZE.
  - DAOBJECT_NAME_W: Retrieves the name of the object in Unicode characters. pInfo points to a buffer of 16 bit characters of size DA_PATHSIZE.
  - DAOBJECT_FORMATID: Retrieves the file ID of the object. pInfo points to a VTDWORD value.
  - DAOBJECT_COMPRESSIONTYPE: Retrieves an identifier of the type of compression used to store the object, if known. pInfo points to a VTDWORD value.

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

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

Return Values

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

4.13 DAGetTreeCount

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

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

Parameters

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

Return Value

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

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

4.14 DAGetTreeRecord

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

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

Parameters

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

Return Values

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

4.14.1 SCCDATREENODE Structure

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

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

Parameters

- dwSize: Must be set by the OEM to sizeof(SCCDATREENODE).
- dwNode: The number of the record to retrieve information about. The first node is node 0.

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

4.15 DAOpenTreeRecord

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

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

`lphDoc` is *not* a file handle.

Parameters

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

Return Value

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

4.16 DAOpenRandomTreeRecord

This function is called to open a record within an archive file and make it accessible by one or more of the data access technologies. It is similar to DAOpenTreeRecord, except that instead of reading the data for all nodes in the archive in a sequential order, this function will only read the data for the requested nodes from the archive. To use this function, you must first process the archive with Content Access or Search Export and save the Node Locator data for later use in this function.

```
DAERR DAOpenRandomTreeRecord(
      VTHDOC      hDoc,
      VTLPHDOC    lphDoc,
      SOTREENODELOCATOR sTreeNodeLocator )
```

lphDoc is not a file handle.

Parameters

- hDoc: Identifier of open document. This hDoc must come from an archive document opened with DAOpenDocument with the flag DAOPENDOCUMENT_ARCHIVEONLYMODE set.
- lphDoc: Pointer to a handle that will be filled with a value that uniquely identifies the document to data access. The developer will use this handle in subsequent calls to data access to identify this particular document.
- sTreeNodeLocator: An SOTREENODELOCATOR structure which contains data identifying the desired node. This data should come from a previous conversion of the archive document using Content Access or Search Export.

Return Value

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

4.16.1 DATREENODELOCATOR

```
typedef struct DATREENODELOCATORtag
{
    VTDWORD dwSize;        /* size of this structure */
    VTDWORD dwSpecialFlag;    /* special flags coming from CA or SX */
    VTDWORD dwData1;        /* dwData1 coming from CA or SX */
    VTDWORD dwData2;        /* dwData2 coming from CA or SX */
}SCCDATREENODELOCATOR, *PSCCDATREENODELOCATOR;
```

4.16.2 SCCCA_TREENODELOCATOR: Tree Node Locator

This content type contains information to be used in the SOTREENODELOCATOR structure, which is used by DAOpenRandomTreeRecord and DASaveRandomTreeRecord.

SCCCA_TREENODELOCATOR Content Description

- dwType: SCCCA_TREENODELOCATOR

- dwSubType: Reserved
- dwData1: SOTREENODELOCATOR.dwSpecialFlags
- dwData2: SOTREENODELOCATOR.dwData1
- dwData3: SOTREENODELOCATOR.dwData2
- dwData4: Reserved
- pDataBuf: Not used

4.17 DASaveInputObject

This function saves a copy of the document or object pointed to by hDoc. The object may be an embedded object, a linked object or a compressed file.

> **Note:**
>
> Some file formats store only partial files as embedded objects. Outside In will not be able to create readable files from these objects. It is recommended that you use DAGetObjectInfo with dwInfoId set to DAOBJECT_FLAGS to discern which objects Outside In will be able to successfully extract.

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

Parameters

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

Return Values

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

4.18 DASaveTreeRecord

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

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

Parameters

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

Return Values

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

Otherwise, one of the other DAERR_ values in sccda.h is returned.

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

4.19 DASaveRandomTreeRecord

This function is called to extract a record in an archive file to disk. It is similar to DASaveTreeRecord, except that instead of reading the data for all nodes in the archive in a sequential order, this function will only read the data for the requested nodes from the archive. To use this function, you must first process the archive with Content Access or Search Export and save the Node Locator data for later use in this function.

```
DAERR DASaveRandomTreeRecord(
      VTHDOC            hDoc,
      SOTREENODELOCATOR sTreeNodeLocator,
      VTDWORD           dwSpecType,
      VTLPVOID          pSpec,
      VTDWORD           dwFlags)
```

Parameters

- hDoc: Identifier of open document. This hDoc must come from an archive document opened with DAOpenDocument with the flag DAOPENDOCUMENT_ARCHIVEONLYMODE set.
- sTreeNodeLocator: An SOTREENODELOCATOR structure which contains data identifying the desired node. This data should come from a previous conversion of the archive document using Content Access or Search Export.
- dwSpecType: Describes the contents of pSpec. Together, dwSpecType and pSpec describe the location of the source file to which the file will be extracted. Must be one of the following values:
  - IOTYPE_ANSIPATH: Windows only. pSpec points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) filename conventions.
  - IOTYPE_REDIRECT: Specifies that redirected I/O will be used to save the file.
  - IOTYPE_UNICODEPATH: Windows only. pSpec points to a NULL-terminated full path name using the Unicode character set and NTFS (Win32 and Win64) file name conventions.
  - IOTYPE_UNIXPATH: X Windows on UNIX platforms only. pSpec points to a NULL-terminated full path name using the system default character set and UNIX path conventions. Unicode paths can be accessed on UNIX platforms by using a UTF-8 encoded path with IOTYPE_UNIXPATH.
- pSpec: File location specification
- dwFlags: Currently not used. Should be set to 0.

Return Value

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

4.19.1 DATREENODELOCATOR

```
typedef struct DATREENODELOCATORtag
{
    VTDWORD dwSize;        /* size of this structure */
    VTDWORD dwSpecialFlag;    /* special flags coming from CA or SX */
    VTDWORD dwData1;        /* dwData1 coming from CA or SX */
    VTDWORD dwData2;        /* dwData2 coming from CA or SX */
}SCCDATREENODELOCATOR, *PSCCDATREENODELOCATOR;
```

4.19.2 SCCCA_TREENODELOCATOR: Tree Node Locator

This content type contains information to be used in the SOTREENODELOCATOR structure, which is used by DAOpenRandomTreeRecord and DASaveRandomTreeRecord.

SCCCA_TREENODELOCATOR Content Description

- dwType: SCCCA_TREENODELOCATOR
- dwSubType: Reserved
- dwData1: SOTREENODELOCATOR.dwSpecialFlags
- dwData2: SOTREENODELOCATOR.dwData1
- dwData3: SOTREENODELOCATOR.dwData2
- dwData4: Reserved
- pDataBuf: Not used

4.20 DACloseTreeRecord

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

```
DAERR DACloseTreeRecord(
      VTHDOC      hDoc);
```

Parameters

- hDoc: Identifier of open record document.

Return Value

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

4.21 DASetStatCallback

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

Use of the Status Callback Function

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

The status callback function has two return values defined:

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

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

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

    return (OIT_STATUS_CONTINUE);
}
```

Prototype

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

Parameters

- pCallback: Pointer to the callback function.
- dwID: Handle that indicates the callback status.
  - OIT_STATUS_WORKING
  - OIT_STATUS_CONTINUE
  - OIT_STATUS_ABORT
- pCallbackData: Currently always NULL

Return Values

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

4.22 DASetFileAccessCallback

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

Use of the File Access Callback

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

The status callback function has two return values defined:

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

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

Prototype

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

Parameters

- pCallback: Pointer to the callback function.

Return Values

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

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

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

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

```
typedef struct {
      VTDWORD   dwSize;          /* size of this structure */
      VTWORD    wFIId;           /* FI id of reference file */
      VTDWORD   dwSpecType;      /* file spec type */
```

```
        VTVOID   *pSpec;            /* pointer to a file spec */
        VTDWORD   dwRootSpecType;   /* root file spec type */
        VTVOID   *pRootSpec;        /* pointer to the root file spec */
        VTDWORD   dwAttemptNumber;  /* The number of times the callback has */
                                    /* already been called for the currently */
                                    /* requested item of information */
} IOREQUESTDATA, * PIOREQUESTDATA;
```

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

> **Note:**
>
> Not all formats that use passwords are supported. *DASetFileAccessCallback* applies to filters that support password protected files. Check filter for any or all calls to *UTGetFileAccess* in filters and core modules.
>
> Only Microsoft Office binary (97-2003), Microsoft Office 2010-2013, Microsoft Outlook PST 97-2016, Lotus NSF, PDF (with RC4 encryption), and 7zip (with AES 128 & 256 bit, ZipCrypto) are currently supported.
>
> Passwords for PST/OST files must be in the Windows single-byte character set. For example, Cyrillic characters should use the 1252 character set. For PST/OST files, Unicode password characters are not supported.

4.23 DAOpenNextDocument

Allows an existing Export or Data Access document handle to be used or reused when opening a new document, enabling options to be preserved across multiple exports, or allowing multiple documents to be exported to the same output destination.

This function uses an existing "reference" handle as a starting point for opening another document. The reference handle may be either a document handle (obtained through DAOpenDocument) or an export handle (obtained via a call to EXOpenExport). The difference between using these two handle types is that certain document specification types (subdocuments of the original document) will not be successfully opened when a document handle is used as the reference handle. If an Export handle is used as the reference handle, subdocument specifications are allowed.

Since the same handle is used multiple times, only a single call to DACloseDocument is needed. Each document is actually closed when the next document is opened; successive calls to DAOpenNextDocument free the resources used in previous calls.

Using this function allows developers to make multiple calls to the EX functions, without having to re-set options every time. Options can be set once for the original document, and retained for each subsequent document.

Additionally, some export libraries allow exporting multiple source documents to a single output document. Currently, this is supported for PDF and multi-page TIFF

output only. To do this, a developer would export the first document normally, then call DAOpenNextDocument to open the subsequent source documents, followed by a call to EXRunExport. EXOpenExport and EXCloseExport should only be called once each for this type of export.

Prototype

```
DAERR DAOpenNextDocument(
     VTHANDLE hReference,
     VTDWORD  dwSpecType,
     VTLPVOID pSpec,
     VTDWORD  dwFlags );
```

Parameters

- hReference: this VTHANDLE value may be either an hDoc, the VTHDOC document handle obtained through a prior call to DAOpenDocument; or an hExport, the VTHEXPORT handle obtained from a prior call to EXOpenExport. This is not an operating system file handle.
- dwSpecType: Describes the contents of pSpec. The dwSpecType values allowed by DAOpenDocument for this parameter are acceptable, with the exceptions that IOTYPE_ARCHIVEOBJECT and IOTYPE_LINKEDOBJECT are only allowed when hReference is an Export handle, obtained via a call to EXOpenExport.
- pSpec: File location specification.
- dwFlags: The low WORD is the file ID for the document (0 by default). If you set the file ID incorrectly, the technology fails. If set to 0, the file identification technology determines the input file type automatically. The high WORD should be set to 0.

Return Values

- DAERR_OK: Returned if the open was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_FEATURENOTAVAIL: Returned if the value specified by dwSpecType is not one of the supported spec types for this operation.

4.24 DAGetOptionItem

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

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

- hDoc: Handle to current document, as described in documentation of DAGetOption.
- dwOptionId: The option id of an option whose item values are being requested.
- dwWhichItem: Must be one of the following:
  - SCCOPT_FIRSTITEM - which retrieves the first item in the specified list.
  - SCCOPT_NEXTITEM - which retrieves the next item in the specified list.

  - SCCOPT_ITEMSIZE - which does not retrieve an item, but sets *pSize to the necessary buffer size for an item of the specified list. (If the list items vary in size, it will indicate the size of the largest item.)
  - .SCCOPT_LISTSIZE - which does not retrieve an item, but sets *pSize to the count of all items in the specified list
- pValue: Pointer to a buffer that will receive the requested item's value.
- pSize: Size of the buffer pointed to by pValue. If this size isn't sufficient to receive the requested data, the function will set the value pointed to by pSize to the size required to hold the item's data, return SCCERR_INSUFFICIENTBUFFER.
- pItemId: Points to a DWORD that will receive the value of an internal identifier of the item. This identifer may be used in subsequent calls to DARemoveOptionItem. This parameter may be NULL if the caller is not interested in the id.

4.25 DARemoveOptionItem

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

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

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

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

4.26 DAAddOptionItem

This function adds a new item to the end of the list of items associated with a mult-value option. The value of the item id is determined internally by the Outside In code, and will not change for the lifetime of the option item.If the caller is not interested in the value of the itemId, calling DASetOption multiple times is equivalent to calling DAAddOptionItem with the pItemId set to NULL. (This only applies to options whose values may be set through DAAddOptionItem).

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

- hDoc: Handle to current document, as described in documentation of DASetOption.
- dwOptionId: The option id of an option that requires a list of values.
- pValue: Pointer to the value of the option item being added.

- dwSize: Size of the data pointed to by pValue.
- pItemId: Points to a DWORD that will receive the value of an internal identifier of the item. This identifer may be used in subsequent calls to DARemoveOptionItem. This parameter may be NULL if the caller is not interested in the id.

4.27 DASetFileSpecOption

This function is called to set the value of an option that takes a spec and spec type as parameters. It is currently only implemented for use in setting the template option in HTML Export. This function only needs to be used if the developer wishes to use Redirected IO on the template files. It may be used to set the template option even if the developer does not wish to use redirected IO, although DASetOption may also be used in this situation.

Prototype

```
DAERR DASetFileSpecOption(
   VTHDOC     hDoc,
   VTDWORD    dwOptionId,
   VTDWORD    dwSpecType,
   VTLPVOID   pSpec);
```

Parameters

- hDoc: Identifier of open document. May be a VTHDOC returned by the DAOpenDocument function, or the subhandle returned by the DAOpenDocument or DAOpenTreeRecord functions (VTHCONTENT, VTHTEXT, etc.). Setting an option for a VTHDOC affects all subhandles opened under it, while setting an option for a subhandle affects only that handle.
- dwOptionId: The identifier of the option to be set. Currently only implemented for the option SCCOPT_EX_TEMPLATE.
- dwSpecType: The spec type of the file. Should be set to one of the valid spec types.
- pSpec: File location specification.

Return Value

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

4.28 DAOpenSubdocumentById

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

Prototype

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

Parameters

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

5
Text Access Functions

The Text Access module is required to use these functions.
This chapter includes the following sections:

5.1 TAOpenText

TAOpenText is used to initiate text access for a file that has been opened by DAOpenDocument.

Prototype

```
DAERR TAOpenText(
   VTHDOC     hDoc,
   VTLPHTEXT  phText )
```

phContent is *not* a file handle.

Parameters

- hDoc: A handle that identifies the document, created by DAOpenDocument.
- phText: Pointer to a handle that will receive a value uniquely identifying the document to the Text Access routines. If the function fails, this value will be set to VTHDOC_INVALID.

Return Values

- DAERR_OK: Open was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.
- DAERR_EMPTYFILE: Empty file.
- DAERR_PROTECTEDFILE: Password protected or encrypted file.
- DAERR_SUPFILEOPENFAILS: Supplementary file open failed.
- DAERR_FILTERNOTAVAIL: The file's type is known, but the appropriate filter is not available.
- DAERR_FILTERLOADFAILED: An error occurred during the initialization of the appropriate filter.

5.2 TACloseText

TACloseText is called to terminate text access for a file.

Prototype

```
DAERR TACloseText(
   VTHTEXT   hText )
```

Parameters

- hText: Text Access handle for the document. Must be a handle returned by the TAOpenText function.

Return Values

- DAERR_OK: Close was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.

5.3 TAReadFirst

This function is called to set the read pointer to the beginning of the document and to retrieve the first block of text.

Prototype

```
DAERR TAReadFirst(
   VTHTEXT    hText,
   VTLPBYTE   pTextBuf,
   VTDWORD    dwBufSize,
   VTLPDWORD  pBufCount )
```

Each piece of content has a type and a subtype. Based on the type and subtype, the content is described by using up to four VTDWORDs and a data buffer provided by the caller. The hText, pTextBuf, dwBufSize, and pBufCount elements of this structure should be filled by the caller before calling TAReadNext or TAReadFirst.

Parameters

- hText: Text Access handle for the document. Must be a handle returned by the TAOpenText function.
- pTextBuf: Pointer to a buffer to receive the first block of text. NULL characters are included in the text buffer to act as fillers for text which was in the original file but is not part of the document body (revision deletions and document properties). Special characters are manufactured by the technology due to special formatting attributes. For more information, see TAReadNext.
- dwBufSize: Size of the buffer pointed to by pTextBuf.
- pBufCount: Pointer to a DWORD that will receive the actual size of the data copied into pTextBuf. Note that for DBCS and Unicode character sets, this will not necessarily be the character count.

Return Values

- DAERR_OK: The read was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.

5.4 TAReadNext

TAReadNext is called to retrieve the next block of text from the file, beginning at the location where the last call to TAReadNext or TAReadFirst ended.

Prototype

```
DAERR TAReadNext(
   VTHTEXT    hText,
   VTLPBYTE   pTextBuf,
   VTDWORD    dwBufSize,
   VTLPDWORD  pBufCount )
```

Each piece of content has a type and a subtype. Based on the type and subtype, the content is described by using up to four VTDWORDs and a data buffer provided by the caller. The hText, pTextBuf, dwBufSize, and pBufCount elements of this structure should be filled by the caller before calling TAReadNext or TAReadFirst.

Parameters

- hText: Text Access handle for the document. Must be a handle returned by the TAOpenText function.
- pTextBuf: Pointer to a buffer to receive the block of text. NULL characters are included in the text buffer to act as fillers for text which was in the original file but is not part of the document body (revision deletions and document properties). Special characters are manufactured by the technology due to special formatting attributes.
- dwBufSize: This is the size of the buffer pointed to by pTextBuf.
- pBufCount: Pointer to a DWORD that will receive the actual size of the data copied into pTextBuf. Note that for DBCS and Unicode character sets, this will not necessarily be the character count.

Return Values

- DAERR_OK: The read was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_EOF: Read was successful, and the end of the file was encountered.
- DAERR_ABORT: A fatal error has occurred, read process was aborted.

Special Text Character Substitutions

- Email Delimiter: 0x09
- End of Database Record: 0x0A
- End of File: 0x0D
- End of Paragraph: 0x0D

- End of Table Cell: 0x0D
- End of Table Row: 0x0D
- Hard Hyphen: 0x2D
- Hard Line Break: 0x0A
- Hard Page Break: 0x0C
- Hard Space: 0x20
- Section Separator: 0x0D
- Syllable Hyphen: 0x2D
- Tab: 0x09
- Word Delimiter: 0x20

6
Content Access Functions

The Content Access module is required to use these functions.
This chapter includes the following sections:

6.1 CAOpenContent

CAOpenContent is used to initiate content access for a file that has been opened by DAOpenDocument.

Prototype

```
DAERR CAOpenContent(
   VTHDOC         hDoc;
   VTLPHCONTENT   phContent;
)
```

`phContent` is *not* a file handle.

Parameters

- hDoc: A handle that identifies the document, created by DAOpenDocument.
- phContent: Pointer to a handle that will receive a value uniquely identifying the document to the Content Access routines. If the function fails, this value will be set to VTHDOC_INVALID.

Return Values

- DAERR_OK: Open was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.
- DAERR_EMPTYFILE: Empty file.
- DAERR_PROTECTEDFILE: Password protected or encrypted file.
- DAERR_SUPFILEOPENFAILS: Supplementary file open failed.
- DAERR_FILTERNOTAVAIL: The file's type is known, but the appropriate filter is not available.

- DAERR_FILTERLOADFAILED: An error occurred during the initialization of the appropriate filter.

6.2 CACloseContent

CACloseContent is called to terminate content access for a file.

Prototype

```
DAERR CACloseContent(
   VTHCONTENT   hContent;
)
```

Parameters

- hContent: Content Access handle for the document. Must be a handle returned by the CAOpenContent function.

Return Values

- DAERR_OK: Close was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.

6.3 CAReadFirst

This function is called to set the read pointer to the beginning of the document content and to obtain the file identification property for the document.

Prototype

```
DAERR CAReadFirst(
   VTHCONTENT         hContent;
   PSCCCAGETCONTENT   pGetContent;
)
```

Parameters

- hContent: Content Access handle for the document. Must be a handle returned by the CAOpenContent function.
- pGetContent: Pointer to a structure of type SCCCAGETCONTENT (see SCCCAGETCONTENT Structure). CAReadFirst will always fill this structure with the file identification property.

Return Values

- DAERR_OK: Read was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_BADPARAM: One of the function parameters was invalid.

6.4 CAReadNext

CAReadNext is called to retrieve text and properties from a file, beginning at the location where the last content was provided.

Prototype

```
DAERR CAReadNext(
   VTHCONTENT         hContent;
   PSCCCAGETCONTENT   pGetContent;
)
```

Parameters

- hContent: Content Access handle for the document. Must be a handle returned by the CAOpenContent function.
- pGetContent: Pointer to a structure of type SCCCAGETCONTENT (see SCCCAGETCONTENT Structure).

Return Values

- DAERR_OK: Read was successful. Otherwise, one of the other DAERR_ values in sccda.h or one of the SCCERR_ values in sccerr.h is returned.
- DAERR_EOF: Read was successful, and the end of the file was encountered.
- DAERR_ABORT: A fatal error has occurred, read process was aborted.

6.4.1 SCCCAGETCONTENT Structure

```
typedef struct SCCCAGETCONTENTtag
   {
   VTDWORD         dwStructSize;
   VTDWORD         dwFlags;
   VTDWORD         dwMaxBufSize;
   VTVOID       *  pDataBuf;
   VTDWORD         dwType;
   VTDWORD         dwSubType;
   VTDWORD         dwData1;
   VTDWORD         dwData2;
   VTDWORD         dwData3;
   VTDWORD         dwData4;
   VTDWORD         dwDataBufSize;
   } SCCCAGETCONTENT, * PSCCCAGETCONTENT;
```

Each piece of content has a type and a subtype. Based on the type and subtype, the content is described by using up to four VTDWORDs and a data buffer provided by the caller. The dwStructSize, dwFlags, pDataBuf and dwMaxBufSize elements of this structure should be filled by the caller before calling CAReadNext or CAReadFirst.

- dwStructSize: Initialized by caller to sizeof(SCCCAGETCONTENT).
- dwFlags: Set by caller. Currently undefined, must be set to 0.
- dwMaxBufSize: Initialized by caller to the size of the buffer pointed to by pDataBuf.
- pDataBuf: This pointer must be set by the caller to a buffer at least 1K in size and must be properly byte-aligned (4 bytes). This buffer will be filled with content information based on dwType.
- dwType: Returns one of the following values (For detailed descriptions of the content types, see Content Description):
  - SCCCA_ANNOTATION: Marks the location of an annotation or sub-document.

  - SCCCA_BEGINTAG: Marks the beginning of a tagged section of the document
  - SCCCA_BREAK: Signals the end of document properties
  - SCCCA_ENDTAG: Marks the end of a tagged section of the document
  - SCCCA_FILEPROPERTY: File identification information
  - SCCCA_GENERATED: Text generated from non-character data
  - SCCCA_OBJECT: Embedded object information
  - SCCCA_SHEET: The name of a sheet in a spreadsheet or presentation
  - SCCCA_STYLECHANGE: Indicates a change in style information
  - SCCCA_TEXT: Normal stream text
  - SCCCA_TREENODELOCATOR: Used by DAOpenRandomTreeRecord and DASaveRandomTreeRecord.
- dwSubType Returns additional information based on `dwType`. Here are some valid subtypes:
  - SCCCA_ANNOTATION: Subtype are values like SCCCA_ANNOTATION_FOOTNOTE or SCCCA_ANNOTATION_ENDNOTE. dwData1 links to the corresponding SCCCA_BEGINTAG.
  - SCCCA_HIDDEN: A valid subtype for the SCCCA_TEXT type representing hidden text.
  - SCCCA_FRAME_EX: A valid subtype for the SCCCA_BEGINTAG and SCCCA_ENDTAG types representing extended frames.
  - SCCCA_LINKEDOBJECT: A valid subtype for the SCCCA_BEGINTAG and SCCCA_ENDTAG types representing an object accessible via a link. When dwSubType equals SCCCA_LINKEDOBJECT, dwData1, dwData2, dwData3 and dwData4 will contain values that are used to locate the object.
  - SCCCA_OCE: A valid subtype for the SCCCA_STYLECHANGE type, indicating a change in the character set in the original document. dwData1 returns the new character set.
- dwData*n*: Return additional information based on dwType and dwSubType. Several examples are shown above.
- dwDataBufSize: Returns the actual size of the data placed in the buffer pointed to by pDataBuf.

6.5 CAContentStatus

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

Prototype

```
DA_ENTRYSC DAERR DA_ENTRYMOD CAContentStatus(VTHCONTENT hContent, VTDWORD
dwStatusType, VTLPVOID pStatus);
```

Parameters

- hContent: Content handle for the document.

- dwStatusType: Specifies which status information should be filled in pStatus.
  - SCCCA_STATUS_INFORMATION - fills in the SCCCASTATUSINFORMATION structure.
- pStatus: A pointer to a SCCCASTATUSINFORMATION data structure

Return Values

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

6.5.1 EXSUBDOCSTATUS Structure

The SCCCASTATUSINFORMATION is defined to be the same as EXSTATUSINFORMATION, which is defined as follows:

```
typedef struct EXSTATUSINFORMATIONtag
{
    VTDWORD dwVersion; /* version of this structure */
    VTBOOL bMissingMap; /* a PDF text run was missing the toUnicode table */
    VTBOOL bVerticalText; /* a vertical text run was present */
    VTBOOL bTextEffects; /* a run that had unsupported text effects applied. One
example is Word Art*/
    VTBOOL bUnsupportedCompression; /* a graphic had an unsupported compression */
    VTBOOL bUnsupportedColorSpace; /* a graphic had an unsupported color space */
    VTBOOL bForms; /* a sub documents had forms */
    VTBOOL bRightToLeftTables; /* a table had right to left columns */
    VTBOOL bEquations; /* a file had equations*/
    VTBOOL bAliasedFont; /* The desired font was missing, but a font alias was used*/
    VTBOOL bMissingFont; /* The desired font wasn't present on the system */
    VTBOOL bSubDocFailed; /* a sub document was not converted */
    VTBOOL bTypeThreeFont; /* a PDF Type 3 embedded font was encountered */
    VTBOOL bUnsupportedShading; /* a PDF input file had an unsupported shading type
*/
    VTBOOL bInvalidHTML; /* invalid HTML was encountered */
    VTBOOL bVectorObjectLimit; /* The vector object limit was reached */
    VTBOOL bInvalidAnnotationNotApplied; /* Annotation/Redaction wasn't displayed */
    VTBOOL bInlineImageFound; /* An inline image was found and may not have been
rendered */
} EXSTATUSINFORMATION;
```

> **Note:**
>
> When processing a document, Content Access never uses fonts, so bAliasedFont and bMissingFont will never report TRUE.
>
> bVectorObjectLimit applies only to WebView Export, and bInvalidAnnotationNotApplied applies only to Image Export, PDF Export, and Web View Export.

6.6 CASeek

Move to a position in CA.

Prototype

```
DA_ENTRYSC DAERR DA_ENTRYMOD CASeek( VTHCONTENT hContent,  PSCCDAPOS pPos )
```

Parameters

- hContent: CA Content returned from CAOpenContent.
- pPos: Pointer to be filled with our position information.

Return Values

DAERR_BADPARAM SCCERR_OK: One of the function parameters was invalid.

6.7 CATell

Get position of CA.

Prototype

```
DA_ENTRYSC DAERR DA_ENTRYMOD CATell ( VTHCONTENT hContent,  PSCCDAPOS pPos )
```

Parameters

- hContent: CA Content returned from CAOpenContent.
- pPos: Pointer to be filled with our position information.

Return Values

DAERR_BADPARAM SCCERR_OK: One of the function parameters was invalid.

7

Content Description

This chapter discusses tagged content and other content topics.
This chapter includes the following sections:

7.1 SCCCA_BEGINTAG/SCCCA_ENDTAG: Tagged Content

The SCCCA_BEGINTAG and SCCCA_ENDTAG content types are used to tag or delimit other content for a particular purpose. This can be especially useful when searching for specific document property values like the author or title of a document. It can also be used to separate subdocument text like headers, footers, and footnotes from the main document text. Tagged text may be nested inside other tagged text, and tags may overlap each other.

Though most tag types are not particularly useful to developers, the Data Access technology provides all of the tag types rather than make a judgment as to usability. Each is briefly described below.

7.1.1 SCCCA_BEGINTAG Content Description

This section lists the applicable parameters and corresponding values.

- dwType
  - SCCCA_BEGINTAG: Beginning of tagged content
  - SCCCA_ENDTAG: End of tagged content
- dwSubType: Tag type - see Tag Types
- dwData1: Additional ID - see Tag Types for more information.
- dwData2: Not used
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Not used

7.1.2 Tag Types

This section lists the applicable values and corresponding descriptions.

- SCCCA_ALTFONTDATA: Reserved
- SCCCA_ANNOTATIONREFERENCE: Tags content that references an annotation
- SCCCA_BOOKMARK: Delimits content tagged as a bookmark
- SCCCA_CAPTIONTEXT: Tags content that is used as a caption on objects such as tables, equations and figures
- SCCCA_CHARACTER: Reserved
- SCCCA_COMPILEDFIELD: Tags content resulting from an application compiling a field code such as a date. The lack of consistent support by applications for this field makes it unreliable as a search property.
- SCCCA_CONDITIONALSTYLE: Reserved
- SCCCA_COUNTERFORMAT: Reserved
- SCCCA_CUSTOMDATAFORMAT: Reserved
- SCCCA_DATEDEFINITION: Reserved
- SCCCA_DIAGRAM: Reserved
- SCCCA_DIAGRAM_*: Reserved
- SCCCA_DOCUMENTPROPERTY: Tags document property content - see Document Property IDs
- SCCCA_DOCUMENTPROPERTYNAME: Name of a user-defined document property (SCCCA_USERDEFINEDPROP)
- SCCCA_EMAILFIELD: Tags fields associated with email formats - see Mail Field IDs

- SCCCA_EMAILFIELDNAME: Tags the name of a non-standard email field.
- SCCCA_EMAILTABLE: Table of email fields
- SCCCA_ENDNOTEREFERENCE: Tags content that references an endnote
- SCCCA_FONTANDGLYPHDATA: Tags content that references font or glyph data
- SCCCA_FOOTER: Delimits content tagged as footer
- SCCCA_FOOTNOTEREFERENCE: Tags content that references a footnote
- SCCCA_FRAME: Tags content stored within a frame
- SCCCA_FRAME_EX: Tags content that references extended frames
- SCCCA_GENERATEDFIELD: Reserved
- SCCCA_GENERATOR: Reserved
- SCCCA_HEADER: Delimits content tagged as header
- SCCCA_HYPERLINK: Delimits content tagged as a hypertext link
- SCCCA_INDEX: Reserved
- SCCCA_INDEXENTRY: Delimits content that should be placed in the index
- SCCCA_INLINEDATAFORMAT: Reserved
- SCCCA_LINKEDOBJECT: Tags content referencing a linked object. These values may change if different options are applied, with different versions of the technology, or after patches are applied.
- SCCCA_LISTENTRY: Reserved
- SCCCA_MERGEENTRY: Reserved
- SCCCA_NAMEDCELLRANGE: Reserved
- SCCCA_REFERENCEDTEXT: Tags text for later reference
- SCCCA_SLIDENOTES: Tags content stored in speaker/slide notes in a presentation document
- SCCCA_SSHEADERFOOTER: Tags content that references headers or footers in spreadsheet files
- SCCCA_STYLE: Delimits a style definition. Styles may contain text, but typically do not. dwData1 is a flag field for SCCCA_STYLE with the value of SCCCA_STYLEFLAG_INLINE_NUMBERING when the style is an inline numbering style.
- SCCCA_SUBDOCPROPERTY: Tags metadata associated with a subdocument, such as a comment. See SCCCA_SUBDOCPROPERTY Document Properties for more information.
- SCCCA_SUBDOCTEXT: Delimits content stored in subdocuments like headers, footers, frames and notes.
- SCCCA_TOA: Reserved
- SCCCA_TOAENTRY: Reserved
- SCCCA_TOC: Reserved
- SCCCA_TOCENTRY: Reserved
- SCCCA_TOF: Reserved

- SCCCA_VECTORSAVETAG: Reserved
- SCCCA_XMPDATA: Document properties parsed out of the XMP data
- SCCCA_XREF: Reserved
- In the following tag types, an asterisk (*) denotes tags that contain revision data which has a sequence ID in dwData1, a User ID in dwData2, and the time (stored as a DOS Date/Time) in dwData3

  SCCCA_SS_REVISIONS container for all of the tracked changes.

  SCCCA_SS_USERNAMES user ID table containing SCCCA_SS_USERNAME tags.

  SCCCA_SS_USERNAME has a user ID and contains SCCCA_REVISION_USER.

  SCCCA_SS_SHEETNAMES sheet table containing SCCCA_SS_SHEETNAME tags.

  SCCCA_SS_SHEETNAME has a sheet ID and contains SCCCA_REVISION_SHEETNAME and text for the name.

  SCCCA_SS_REV_RENAMESHEET * contains a SCCCA_REVISION_SHEET, which contain the new and old sheet ID's.

  SCCCA_SS_REV_CREATE * empty tag used to output User ID and Date/Time of file creation.

  SCCCA_SS_REV_SAVE * empty tag used to output User ID and Date/Time of a save.

  SCCCA_SS_REV_MODIFYCELL *describes a cell that was changed. It contains SCCCA_REVISION_CELL describing the location of the modified cell, a SCCCA_SS_REV_OLDCELLCONTENT tag, and a SCCCA_SS_REV_NEWCELLCONTENT tag.

  SCCCA_SS_REV_MOVECELLS * describes a cell that was moved and contains a SCCCA_SS_REV_OLDCELLLOCATION tag and a SCCCA_SS_REV_NEWCELLLOCATION tag.

  SCCCA_SS_REV_OLDCELLLOCATION describes the original cell location and contains two SCCCA_REVISION_CELL tags indicating the upper left and lower right coordinates.

  SCCCA_SS_REV_NEWCELLLOCATION describes the new cell location and contains two SCCCA_REVISION_CELL tags indicating the upper left and lower right coordinates.

  SCCCA_SS_REV_ADDROW * contains SCCCA_REVISION_ROW denoting row(s) added.

  SCCCA_SS_REV_DELETEROW * contains SCCCA_REVISION_ROW denoting row(s) deleted. May Contain SCCCA_SS_REV_NEWCELL, which contains the cell information deleted within the row.

  SCCCA_SS_REV_INSERTCOL * contains SCCCA_REVISION_COLUMN denoting column(s) added.

  SCCCA_SS_REV_DELETECOL * contains SCCCA_REVISION_COLUMN denoting column(s) deleted. It may optionally contain new cell and formatting records.

SCCCA_SS_REV_NEWCELL * contains SCCCA_REVISION_CELL denoting new cell location. It may optionally contain formatting records, numeric information, or string information.

SCCCA_SS_REV_CLEARCELL * contains SCCCA_REVISION_CELL denoting old cell location. It may optionally contain numeric information or string information.

SCCCA_SS_REV_OLDCELLCONTENT may contain numeric information or string information.

SCCCA_SS_REV_NEWCELLCONTENT may contain numeric information or string information.

SCCCA_SS_REV_ADDSHEET * contains a SCCCA_REVISION_SHEET.

SCCCA_SS_REV_FORMAT * contains formatting information.

When dwSubType is SCCCA_DOCUMENTPROPERTY, dwData1 will be one of the values listed in the header file sccca.h. The following section, Document Property IDs, lists many of the common document property types. Any content generated between the begin and end tag defines the value of the document property.

When dwSubType is SCCCA_EMAILFIELD, dwData1 will be one of the values in Mail Field IDs, and any content generated between the begin and end tag defines the value of the email field.

7.1.3 Document Property IDs

The following is a partial list of document property IDs.

- SCCCA_ABSTRACT
- SCCCA_ACCOUNT
- SCCCA_ADDRESS
- SCCCA_APPVERSION
- SCCCA_ATTACHMENTS
- SCCCA_AUTHORIZATION
- SCCCA_BACKUPDATE
- SCCCA_BASEFILELOCATION
- SCCCA_BILLTO
- SCCCA_BLINDCOPY
- SCCCA_CARBONCOPY
- SCCCA_CATEGORY
- SCCCA_CHECKEDBY
- SCCCA_CLIENT
- SCCCA_COMPANY
- SCCCA_COMPLETEDDATE
- SCCCA_COUNTBYTES
- SCCCA_COUNTCHARS
- SCCCA_COUNTCHARSWITHSPACES

- SCCCA_COUNTLINES
- SCCCA_COUNTMMCLIPS
- SCCCA_COUNTNOTES
- SCCCA_COUNTPAGES
- SCCCA_COUNTPARAS
- SCCCA_COUNTSLIDES
- SCCCA_COUNTSLIDESHIDDEN
- SCCCA_COUNTWORDS
- SCCCA_CREATIONDATE
- SCCCA_DEPARTMENT
- SCCCA_DESTINATION
- SCCCA_DISPOSITION
- SCCCA_DIVISION
- SCCCA_DOCCOMMENT
- SCCCA_DOCNUMBER
- SCCCA_DOCTYPE
- SCCCA_EDITMINUTES
- SCCCA_EDITOR
- SCCCA_FORWARDTO
- SCCCA_GROUP
- SCCCA_HEADINGPAIRS
- SCCCA_KEYWORD
- SCCCA_LANGUAGE
- SCCCA_LASTPRINTDATE
- SCCCA_LASTSAVEDATE
- SCCCA_LASTSAVEDBY
- SCCCA_LINKSDIRTY
- SCCCA_MAILSTOP
- SCCCA_MANAGER
- SCCCA_MATTER
- SCCCA_OFFICE
- SCCCA_OPERATOR
- SCCCA_OWNER
- SCCCA_PRESENTATIONFORMAT
- SCCCA_PRIMARYAUTHOR
- SCCCA_PROJECT
- SCCCA_PUBLISHER

- SCCCA_PURPOSE
- SCCCA_RECEIVEDFROM
- SCCCA_RECORDEDBY
- SCCCA_RECORDEDDATE
- SCCCA_REFERENCE
- SCCCA_REVISIONDATE
- SCCCA_REVISIONNOTES
- SCCCA_REVISIONNUMBER
- SCCCA_SCALECROP
- SCCCA_SECONDARYAUTHOR
- SCCCA_SECTION
- SCCCA_SECURITY
- SCCCA_SOURCE
- SCCCA_STATUS
- SCCCA_SYSTEM_FILECREATED
- SCCCA_SYSTEM_FILEMODIFIED
- SCCCA_SYSTEM_FILESIZE
- SCCCA_SUBJECT
- SCCCA_TITLE
- SCCCA_TITLEOFPARTS
- SCCCA_TYPIST
- SCCCA_USERDEFINEDPROP
- SCCCA_VERSIONDATE
- SCCCA_VERSIONNOTES
- SCCCA_VERSIONNUMBER

> **Note:**
>
> Document Properties with IDs of SCCCA_USERDEFINEDPROP or above are user-defined properties.

7.1.4 SCCCA_SUBDOCPROPERTY Document Properties

The following values are properties of SCCCA_SUBDOCPROPERTY:

- SCCCA_SUBDOC_AUTHOR
- SCCCA_SUBDOC_CREATEDATE
- SCCCA_SUBDOC_LASTSAVEDATE
- SCCCA_SUBDOC_TITLE

- SCCCA_SUBDOC_NOTES
- SCCCA_SUBDOC_AUTHORSHORT

7.1.5 Mail Field IDs

This is a partial list of fields found in mail documents and archives.

- SCCCA_MAIL_ALTERNATE_RECIPIENT_ALLOWED
- SCCCA_MAIL_ATTACHMENT
- SCCCA_MAIL_ATTENDEES
- SCCCA_MAIL_ATTR_HIDDEN
- SCCCA_MAIL_ATTR_READONLY
- SCCCA_MAIL_ATTR_SYSTEM
- SCCCA_MAIL_AUTO_FORWARDED
- SCCCA_MAIL_BCC
- SCCCA_MAIL_CATEGORIES
- SCCCA_MAIL_CC
- SCCCA_MAIL_CCME
- SCCCA_MAIL_CLIENT_SUBMIT_TIME
- SCCCA_MAIL_COMPANY
- SCCCA_MAIL_CONVERSATION_INDEX
- SCCCA_MAIL_CONVERSATION_TOPIC
- SCCCA_MAIL_CREATION_TIME
- SCCCA_MAIL_CREATOR_ENTRYID
- SCCCA_MAIL_CREATOR_NAME
- SCCCA_MAIL_DEFERRED_DELIVERY_TIME
- SCCCA_MAIL_DELETE_AFTER_SUBMIT
- SCCCA_MAIL_EMAIL
- SCCCA_MAIL_ENTRYID
- SCCCA_MAIL_EXPIRES
- SCCCA_MAIL_EXPIRY_TIME
- SCCCA_MAIL_FLAGSTS
- SCCCA_MAIL_FROM
- SCCCA_MAIL_FULLNAME
- SCCCA_MAIL_HOMEPHONE
- SCCCA_MAIL_IMPORTANCE
- SCCCA_MAIL_INET_MAIL_OVERRIDE_FORMAT
- SCCCA_MAIL_INTERNET_ARTICLE_NUMBER
- SCCCA_MAIL_INTERNET_CPID

- SCCCA_MAIL_INTERNET_MESSAGE_ID
- SCCCA_MAIL_JOBTITLE
- SCCCA_MAIL_LASTMODIFIED
- SCCCA_MAIL_LAST_MODIFIER_ENTRYID
- SCCCA_MAIL_LAST_MODIFIER_NAME
- SCCCA_MAIL_LATEST_DELIVERY_TIME
- SCCCA_MAIL_LOCATION
- SCCCA_MAIL_MESSAGE_CLASS
- SCCCA_MAIL_MESSAGE_CODEPAGE
- SCCCA_MAIL_MESSAGE_LOCALE_ID
- SCCCA_MAIL_MESSAGE_SUBMISSION_ID
- SCCCA_MAIL_MSGFLAG
- SCCCA_MAIL_MSG_EDITOR_FORMAT
- SCCCA_MAIL_NEWSGROUPS
- SCCCA_MAIL_NORMALIZED_SUBJECT
- SCCCA_MAIL_NT_SECURITY_DESCRIPTOR
- SCCCA_MAIL_ORIGINATOR_DELIVERY_REPORT_REQUESTED
- SCCCA_MAIL_PRIORITY
- SCCCA_MAIL_PROFILE_CONNECT_FLAGS
- SCCCA_MAIL_RCVD_BY_FLAGS
- SCCCA_MAIL_RCVD_REPRESENTING_ADDRTYPE
- SCCCA_MAIL_RCVD_REPRESENTING_EMAIL_ADDRESS
- SCCCA_MAIL_RCVD_REPRESENTING_ENTRYID
- SCCCA_MAIL_RCVD_REPRESENTING_FLAGS
- SCCCA_MAIL_RCVD_REPRESENTING_NAME
- SCCCA_MAIL_RCVD_REPRESENTING_SEARCH_KEY
- SCCCA_MAIL_READ_RECEIPT_REQUESTED
- SCCCA_MAIL_RECEIVED
- SCCCA_MAIL_RECEIVED_BY_ADDRTYPE
- SCCCA_MAIL_RECEIVED_BY_EMAIL_ADDRESS
- SCCCA_MAIL_RECEIVED_BY_ENTRYID
- SCCCA_MAIL_RECEIVED_BY_NAME
- SCCCA_MAIL_RECEIVED_BY_SEARCH_KEY
- SCCCA_MAIL_RECIPIENT_REASSIGNMENT_PROHIBITED
- SCCCA_MAIL_REPLY_REQUESTED
- SCCCA_MAIL_REPLY_TIME
- SCCCA_MAIL_REPORT_TAG

- SCCCA_MAIL_RESPONSE_REQUESTED
- SCCCA_MAIL_RTFBODY
- SCCCA_MAIL_RTF_IN_SYNC
- SCCCA_MAIL_RTF_SYNC_BODY_COUNT
- SCCCA_MAIL_RTF_SYNC_BODY_CRC
- SCCCA_MAIL_RTF_SYNC_BODY_TAG
- SCCCA_MAIL_RTF_SYNC_PREFIX_COUNT
- SCCCA_MAIL_RTF_SYNC_TRAILING_COUNT
- SCCCA_MAIL_SEARCH_KEY
- SCCCA_MAIL_SENDER_ADDRTYPE
- SCCCA_MAIL_SENDER_EMAIL_ADDRESS
- SCCCA_MAIL_SENDER_ENTRYID
- SCCCA_MAIL_SENDER_FLAGS
- SCCCA_MAIL_SENDER_NAME
- SCCCA_MAIL_SENDER_SEARCH_KEY
- SCCCA_MAIL_SENSITIVITY
- SCCCA_MAIL_SENT_REPRESENTING_ADDRTYPE
- SCCCA_MAIL_SENT_REPRESENTING_EMAIL_ADDRESS
- SCCCA_MAIL_SENT_REPRESENTING_ENTRYID
- SCCCA_MAIL_SENT_REPRESENTING_FLAGS
- SCCCA_MAIL_SENT_REPRESENTING_NAME
- SCCCA_MAIL_SENT_REPRESENTING_SEARCH_KEY
- SCCCA_MAIL_SIZE
- SCCCA_MAIL_SUBJECT
- SCCCA_MAIL_SUBMITTIME
- SCCCA_MAIL_TO
- SCCCA_MAIL_TRANSPORT_MESSAGE_HEADERS
- SCCCA_MAIL_TRUST_SENDER
- SCCCA_MAIL_WEBPAGE
- SCCCA_MAIL_WORKPHONE

7.2 SCCCA_BREAK: Content Breaks

This content type is used internally, and may be ignored.

7.3 SCCCA_CELL: Cell Boundary

SCCCA_CELL will appear before the contents of a cell in a spreadsheet or database and will contain coordinates that indicate the starting and ending position of the cell. If

the cell isn't merged, then the starting and ending positions will be the same. The content contained by the cell is assumed to end when the next SCCCA_CELL or SCCCA_SHEET is output.

7.3.1 SCCCA_CELL Content Description

- dwType: SCCCA_CELL
- dwSubType: Either SCCCA_HIDDEN if the hidden attribute is set on either the row or column for the cell, or 0 if the cell isn't hidden.
- dwData1: The starting row in a numeric format that is 0 based
- dwData2: The starting column in a numeric format that is 0 based
- dwData3: The ending row in a numeric format that is 0 based
- dwData4: The ending column in a numeric format that is 0 based
- pDataBuf: Not used

7.4 SCCCA_COMMENTREFERENCE

A SCCCA_COMMENTREFERENCE is placed in the actual location of the comment. The body of the comment may appear elsewhere and will be tagged with a SCCCA_BEGINTAG of type SCCCA_SUBDOCTEXT and will have the same Id as the SCCCA_COMMENTREFERENCE.

- dwType: SCCCA_COMMENTREFERENCE
- dwSubType: None
- dwData1: Type of the comment reference anchor. SCCCA_COMMENT_PARAGRAPH, SCCCA_COMMENT_CELL, SCCCA_COMMENT_SLIDE, or SCCCA_COMMENT_VECTORPAGE.
- dwData2: id of the associated subdoc
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Not used

7.5 SCCCA_FILEPROPERTY: File Property Content

Returns the file identification information for a document. This property is generated by the CAReadFirst function.

7.5.1 SCCCA_FILEPROPERTY Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_FILEPROPERTY
- dwSubType: SCCCA_FILEID
- dwData1: One of the file identifier values (FI_*) defined in sccfi.h
- dwData2: The input file's initial character set

- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Not used

7.6 SCCCA_GENERATED: Generated Information

Identical to SCCCA_TEXT, except that the characters come not from the original document, but from some other non-character data (numbers in spreadsheets, dates, and so forth). Because the text is not from the original document, the characters do not contribute toward character counts.

7.6.1 SCCCA_GENERATED Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_GENERATED
- dwSubType: Possible values include the following:
  - SCCCA_BOOKMARKTEXT: Text for the internal name of the bookmark.
  - SCCCA_DOCUMENTTEXT: Regular document text is returned with this subtype.
  - SCCCA_REVISIONDELETE: Will be OR-ed with either SCCCA_DOCUMENTTEXT or SCCCA_SPECIALTEXT when text has been deleted from the final version of a document as a result of a revision.
  - SCCCA_URLTEXT: Text for the Link Location part of a URL.
  - SCCCA_XMPMETADATA: Text from embedded XMP metadata.
- dwData1: Number of characters provided in pDataBuf
- dwData2: Original character set of the text in pDataBuf
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Text buffer. Filled with one or more single- or double-byte characters.

7.7 SCCCA_OBJECT: SubObjects

This content type is provided to allow the developer to access the content of SubObjects, like embedded graphics or objects in an archive. The SubObject can then be opened by DAOpenDocument, filling the IOSPECSUBOBJECT or the IOSPECARCHIVEOBJECT parameter with one of the following values:

7.7.1 SCCCA_OBJECT Content Description

These values may change if different options are applied, with different versions of the technology, or after patches are applied.

- dwType: SCCCA_OBJECT

- dwSubType: Set to SCCCA_EMBEDDEDOBJECT (0) if the sub-object is an embedding or is set to the type of node if the object is from an archive. Possible values include the following:
  - SCCCA_EMBEDDEDOBJECT
  - SCCCA_ARCHIVEITEMCONTAINER
  - SCCCA_COMPRESSEDFILE
  - SCCCA_MESSAGE
  - SCCCA_CONTACT
  - SCCCA_CALENDARENTRY
  - SCCCA_NOTE
  - SCCCA_TASK
  - SCCCA_JOURNALENTRY
  - SCCCA_ATTACHMENT
- dwData1: The internal SubObject identifier or a node identifier.
- dwData2: Stream identifier for an alternate graphic.
- dwData3: Stream identifier for an OLE object if one exists. Otherwise, it is CA_INVALIDITEM.
- dwData4: Object Flags. Currently, 0 or SCCCA_ENDRECORD
- pDataBuf: Not used

7.8 SCCCA_OBJECTALTSTRING: Alternate String

This content type provides an alternate string to identify an embedded object.

7.8.1 SCCCA_OBJECTALTSTRING Content Description

- dwType: SCCCA_OBJECTALTSTRING
- dwSubType: Not used
- dwData1: Number of characters provided in pDataBuf
- dwData2: Original character set of the text in pDataBuf
- dwData3: Not used
- dwData4: Not used
- pDataBuf: Text buffer containing the alternate string. Filled with one or more single- or double-byte characters.

7.9 SCCCA_OBJECTNAME: Object Name

This content type is provided to identify the name of an embedded object.

7.9.1 SCCCA_OBJECTNAME Content Description

- dwType: SCCCA_OBJECTNAME

- dwSubType: Not used
- dwData1: Number of characters provided in pDataBuf
- dwData2: Original character set of the text in pDataBuf
- dwData3: Not used
- dwData4: Not used
- pDataBuf: Text buffer containing the name. Filled with one or more single- or double-byte characters.

7.10 SCCCA_RECORD: Archive Record

This content is output to allow the customer to easily group fields that appear in an archive or in an email archive. The record is considered to be open until a SCCCA_OBJECT is encountered with the flag SCCCA_ENDRECORD set.

7.10.1 SCCCA_RECORD Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_RECORD
- dwSubType: Reserved
- dwData1: Reserved
- dwData2: Reserved
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: not used

7.11 SCCCA_REVISION_CELL: Revision Cell

The location of a cell within a track changes block.

7.11.1 SCCCA_REVISION_CELL Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_REVISION_CELL
- dwSubType: Reserved
- dwData1: Sheet
- dwData2: Column
- dwData3: Row
- dwData4: Reserved
- pDataBuf: Reserved

7.12 SCCCA_REVISION_ROW: Revision Row

This describes a series of rows within a track changes block.

7.12.1 SCCCA_REVISION_ROW Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_REVISION_ROW
- dwSubType: Reserved
- dwData1: Sheet
- dwData2: Start Row
- dwData3: End Row (will be the same as Start Row if a single row is selected)
- dwData4: Reserved
- pDataBuf: Reserved

7.13 SCCCA_REVISION_COLUMN: Revision Column

This describes a series of columns within a track changes block.

7.13.1 SCCCA_REVISION_COLUMN Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_REVISION_COLUMN
- dwSubType: Reserved
- dwData1: Sheet
- dwData2: Start Column
- dwData3: End Column (will be the same as Start Column if a single column is selected)
- dwData4: Reserved
- pDataBuf: Reserved

7.14 SCCCA_REVISION_SHEET: Revision Sheet

This describes the new and old sheet names within a track changes block. The numbers will relate to names output with SCCCA_REVISION_SHEETNAME tags.

7.14.1 SCCCA_REVISION_SHEET Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_REVISION_SHEET
- dwSubType: Reserved

- dwData1: Sheet Number
- dwData2: New Name
- dwData3: Old Name
- dwData4: Reserved
- pDataBuf: Reserved

7.15 SCCCA_REVISION_SHEETNAME: Revision Sheet Name

Provides the name and number of a sheet within a track changes block.

7.15.1 SCCCA_REVISION_SHEETNAME Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_REVISION_SHEETNAME
- dwSubType: Reserved
- dwData1: Sheet Number
- dwData2: Reserved
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Name

7.16 SCCCA_REVISION_USER: Revision User

This describes the name associated with a user ID.

7.16.1 SCCCA_REVISION_USER Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_SHEET
- dwSubType: Reserved
- dwData1: User ID
- dwData2: Reserved
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: User Name

7.17 SCCCA_SHEET: Sheet Names

This content type contains only the sheet name (worksheet in a spreadsheet, slide in presentation, and so forth). This content is *not* optional. It is always created if the information is present. Of course, the client can ignore this text when it is returned.

7.17.1 SCCCA_SHEET Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_SHEET
- dwSubType: Reserved
- dwData1: The length of the name in pDataBuf in characters.
- dwData2: The original character set of the name in pDataBuf.
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Points to the sheet name in whatever output character set has been requested.

7.18 SCCCA_SLIDE: Presentation Slide

SCCCA_SLIDE appears before the contents of a slide in a presentation document. The content contained by the slide is assumed to end when the next SCCCA_SLIDE is output, or the end of the document is reached.

- dwType: SCCCA_SLIDE
- dwSubType: Reserved
- dwData1: Identifies if the slide is hidden (SCCCA_SLIDEHIDDEN) or not (SCCCA_SLIDENORMAL)
- dwData2: Reserved
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Reserved

7.19 SCCCA_STYLECHANGE: Style Information

The SCCCA_STYLECHANGE content type is used to indicate changes in style information. This style information can be used to delimit particularly interesting content.

7.19.1 SCCCA_STYLECHANGE Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_STYLECHANGE

- dwSubType: Possible values include the following:
  - SCCCA_PARASTYLE: pDataBuf indicates the name of the style.
  - SCCCA_HEIGHTANDSPACING: When dwSubType is SCCCA_HEIGHTANDSPACING, dwData1 can be SCCCA_HEIGHT (dwData2 represents the new character height), SCCCA_SPACING (dwData3 represents the new line spacing) or both of these values OR-ed together.
  - SCCCA_INDENTS: When dwSubType is SCCCA_INDENTS, dwData1 can be SCCCA_LEFTINDENT (dwData2 represents the left indent), SCCCA_RIGHTINDENT (dwData3 represents the right indent), SCCCA_FIRSTINDENT (dwData4 represents the first line indent), or any of these values OR-ed together.
  - SCCCA_OCE: This content type provides information about the original charsets of the characters that follow. dwData1 represents the charset as defined in vtchars.h.
- dwData1: Depends on the value of dwSubType.
- dwData2: Depends on the value of dwSubType.
- dwData3: Depends on the value of dwSubType.
- dwData4: Depends on the value of dwSubType.
- pDataBuf: Text buffer. Filled with one or more single- or double-byte characters.
- dwDataBufSize: Size of pDataBuf, in bytes.

7.20 SCCCA_TEXT: Text Content

This content type denotes document text, including special characters such as page breaks and tabs.

The technology guarantees that the text generated by the Content Access technology is identical to the text generated by the Outside In Viewer technology raw-text feature. This allows character counts generated at indexing time using Content Access to be directly mapped to viewer positions at viewing time for search-hit highlighting. However, Content Access has abilities beyond the raw-text feature of the Viewer, such as the ability to retrieve non-visible text such as document properties and hidden text, and the ability to retrieve text from embedded documents.

When the output character is DBCS or Unicode, the character count will not be the same as the buffer byte count because these character sets may generate more than one byte per character. The byte ordering used for multi-byte character sets such as these will be system-dependent; on a computer using an Intel processor, the low byte will be first.

It is important to note that generated numeric data fields, such as date, time, and spreadsheet numbers, are not included in the content returned by SCCCA_TEXT. For information on how such text can be returned by Content Access, see SCCCA_GENERATED: Generated Information.

7.20.1 SCCCA_TEXT Content Description

This section lists the applicable parameters and corresponding values.

- dwType: SCCCA_TEXT

- dwSubType: One of the following values:
  - SCCCA_DOCUMENTTEXT: Regular document text is returned with this subtype.
  - SCCCA_SPECIALTEXT: Used to return text elements that are manufactured by the technology due to special formatting attributes.

  SCCCA_DOCUMENTTEXT or SCCCA_SPECIALTEXT can be optionally OR-ed with any of the following to specify the type of text to be returned:
  - SCCCA_ALLCAPS
  - SCCCA_BOLD
  - SCCCA_DUNDERLINE
  - SCCCA_HIDDEN
  - SCCCA_ITALIC
  - SCCCA_OUTLINE
  - SCCCA_REVISIONDELETE: Text that has been deleted from the final version of a document as a result of a revision.
  - SCCCA_REVISIONADD: Text that has been added to the final version of a document as a result of a revision.
  - SCCCA_SMALLCAPS
  - SCCCA_STRIKEOUT
  - SCCCA_UNDERLINE
  - SCCCA_UNKNOWNMAP: This flag is set when PDF files don't contain a ToUnicode map. This indicates that the mappings may or may not be correct.
- dwData1: Number of characters provided in pDataBuf
- dwData2: Original character set of the text in pDataBuf
- dwData3: Reserved
- dwData4: Reserved
- pDataBuf: Text buffer. Filled with one or more single- or double-byte characters.

7.20.2 Special Text Character Substitutions

- Context Change: 0x0D
- Email Delimiter: 0x09
- End of Database Record: 0x0A
- End of File: 0x0D
- End of Paragraph: 0x0D
- End of Table Cell: 0x0D
- End of Table Row: 0x0D
- Hard Hyphen: 0x2D
- Hard Line Break: 0x0A
- Hard Page Break: 0x0C

- Hard Space: 0x20
- Implied Space: 0x20
- Section Separator: 0x0D
- Syllable Hyphen: 0x2D
- Tab: 0x09

7.21 SCCCA_TREENODELOCATOR: Tree Node Locator

This content type contains information to be used in the SOTREENODELOCATOR structure, which is used by DAOpenRandomTreeRecord and DASaveRandomTreeRecord. These values may change if different options are applied, with different versions of the technology, or after patches are applied.

7.21.1 SCCCA_TREENODELOCATOR Content Description

- dwType: SCCCA_TREENODELOCATOR
- dwSubType: Reserved
- dwData1: SOTREENODELOCATOR.dwSpecialFlags
- dwData2: SOTREENODELOCATOR.dwData1
- dwData3: SOTREENODELOCATOR.dwData2
- dwData4: Reserved
- pDataBuf: Not used

8
Redirected IO

This chapter addresses how developers have total control over access to a file via Outside In's redirected IO mechanism.
Many developers using the earlier versions of this technology expressed a need to read file data from non-file system based sources. For instance, the developer might want to read the file from a database on a server. Perhaps the developer is downloading the file over a slow link, and wants to see the first screen of a document before the download is completed, or only wants to download enough to view the first screen.

This chapter includes the following sections:

8.1 Using Redirected IO

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

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

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

Because the pSpec passed is essentially the file handle that the view window uses, the developer can redirect the IO on a file-by-file basis while still viewing regular disk-based files.

The BASEIO structure is defined as follows:

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

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

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

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

8.2 IOClose

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

Prototype

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

Parameters

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

Return Values

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

8.3 IORead

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

Prototype

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

Parameters

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

Return Values

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

8.4 IOWrite

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

> **Note:**
>
> This function has been fully documented only for completeness. OEMs who use redirected IO do not need to implement writing and the IOWrite function should do nothing but return IOERR_UNKNOWN.

Prototype

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

Parameters

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

Return Values

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

8.5 IOSeek

Moves the current file position.

Prototype

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

Parameters

- hFile: Identifies the file to be read. Should be cast into a pointer to your data structure (MYFILE in the preceding discussion).
- wFrom: One of the following values:
  - IOSEEK_TOP: Move the file position lOffset bytes from the top (beginning) of the file.
  - IOSEEK_BOTTOM: Move the file position lOffset bytes from the bottom (end) of the file.
  - IOSEEK_CURRENT: Move the file position lOffset bytes from the current file position.
- lOffset: Number of bytes to move the file pointer. A positive value moves the file pointer forward in the file and a negative value moves it backward. If a requested seek value would move the file pointer before the beginning of the file, the file pointer should remain unchanged and IOERR_UNKNOWN should be returned. Seeking past EOF is allowed. In that case IOERR_OK should be returned. IOTell

would return the requested seek position and IORead should return IOERR_EOF and 0 bytes read.

Return Values

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

8.6 IOTell

Returns the current file position.

Prototype

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

Parameters

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

Return Values

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

8.7 IOGetInfo

Returns information about an open file.

Prototype

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

Parameters

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

- IOGETINFO_PATHNAME: pInfo points to a string that should be filled with the fully qualified path name (including the file name) of the open file. For example, C:\MYDIR\TEST.DOC. If you do not know the path name, return IOERR_UNKNOWN.
- IOGETINFO_PATHTYPE: pInfo points to a DWORD that should be filled with the IOTYPE of the path returned by IOGETINFO_PATHNAME. For instance, if you return a DOS path name in the Unicode character set, you should return IOTYPE_UNICODEPATH.
- IOGETINFO_ISOLE2STORAGE: Must return IOERR_FALSE. pInfo is not used.
- IOGETINFO_GENSECONDARY: pInfo points to a structure of type IOGENSECONDARY. Some file types require supporting files to be opened. These supporting files may contain formatting information or extra data. Correct handling of IOGETINFO_GENSECONDARY is critical to the operation of the Outside In technology. For a list of these file types, see File Types That Cause IOGETINFO_GENSECONDARY.

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

  The developer may either use one of the standard Outside In IO types or totally redirect the IO for the secondary file, as well. For more details, see IOGENSECONDARY and IOGENSECONDARYW Structures.
- IOGETINFO_64BITIO: For redirected I/O that wishes to use 64-bit seek/tell functions, your IOGetInfo function must respond IOERR_TRUE to this dwInfoId. In addition, the pSeek64/pTell64 items in the baseio structure must be valid pointers to the proper function types.
- IOGETINFO_DPATHNAME: pInfo points to a structure of type DPATHNAME, which should be filled with the fully qualified path name (including the file name) of the open file, for example, C:\MYDIR\TEST.DOC. If you do not know the path name, return IOERR_UNKNOWN. The dwPathLen element contains the size of the buffer pointed to by the pPath element. If the buffer size is too small to contain the full path, modify dwPathLen to be the correct size of the buffer required to hold the path name in its IOTYPE character width including the NULL terminator and return IOERR_INSUFFICIENTBUFFER.

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

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

Parameters

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

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

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

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

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

Parameters

dwSize: Will be set to sizeof (IOGENSECONDARYDP)

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

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

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

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

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

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

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

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

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

Any other value should return IOERR_BADINFOID.

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

Return Values

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

8.7.1 IOGENSECONDARY and IOGENSECONDARYW Structures

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

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

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

```
typedef struct
{
   VTDWORD     dwSize;
   VTLPBYTE    pFileName;
   VTDWORD     dwSpecType;
   VTLPVOID    pSpec;
   VTDWORD     dwOpenFlags
} IOGENSECONDARY, * PIOGENSECONDARY;

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

- dwSize: Will be set to sizeof (IOGENSECONDARY) or sizeof (IOGENSECONDARYW) (both of these values are the same).
- pFileName: A pointer to a string representing the file name of the secondary file that the technology requires. It will generally be a name that is stored in the primary file somewhere (such as MYSTYLE.STY for a Word for DOS file) or a name generated from the primary file name (the primary file for a DataEase

database will always have a .dba extension, the secondary name would be the same file name but with a .dbm extension).

- dwSpecType: The developer must fill this with the IOSPEC for the secondary file.
- pSpec: On entry, this pointer points to an array of 1024 bytes. If the dwSpecType is set a regular IOTYPE such as IOTYPE_ANSIPATH, the developer may fill this array with the path name or structure required for that IOTYPE. If the developer is redirecting access to the secondary file, then dwSpecType will be IOTYPE_REDIRECT and the developer should replace pSpec with a pointer to a developer-defined structure that begins with the BASEIO structure (see Using Redirected IO).

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

8.7.2 File Types That Cause IOGETINFO_GENSECONDARY

The following details concern specific file types.

- Microsoft Word for DOS Versions 4, 5 and 6: Used to open and read the style sheet file associated with the document. The filter will successfully degrade if the style sheet is not present.
- Harvard Graphics DOS 3.x: Used to open and read the individual slides within ScreenShow and palette files. Files with the extension .ch3 are individual graphics or slides that can be opened using no secondary files. Files with the extension .sy3 are ScreenShows that reference a list of .ch3 files via the secondary file mechanism. There is also an optional palette file that can be referenced from a .ch3 file, but the filter will successfully degrade if the palette file is not present.
- R:Base: Used to open and read required schema file. The R:Base data files are named *xxxx*2.rbf but the data is useless without the schema file named *xxxx*1.rbf. There is also a *xxxx*3.rbf file associated with each database, but it is not used.
- Paradox 4.0 and Above: Used to open and read memo field data file. Paradox uses a separate file for all memo field data larger than 32 bytes.
- DataEase: Used to open and read the data file. DataEase databases include a .dba file that contains the schema (the file that the technology can identify as DataEase) and a .dbm file that contains the actual data.

8.8 IOSEEK64PROC / IOTELL64PROC

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

8.8.1 IOSeek64

Moves the current file position.

Prototype

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

Parameters

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

8.8.2 IOTell64

Returns the current file position.

Prototype

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

Parameters

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

9 Implementation Issues

This chapter discusses potential issues in using Content Access.

9.1 Running in 24x7 Environments

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

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

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

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

10
Sample Applications

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

This chapter includes the following sections:

10.1 Building the Samples on a Windows System

Microsoft Visual Studio project files are provided for building each of the sample applications. For 32-bit versions of Windows, versions of the project files are provided for Visual Studio 2013 (.dsp files) and Visual Studio 2013 (.vcproj files).

> **Note:**
>
> Because .vcproj files may not pick up the right compiler on their own, you need to make sure that you are building with the Win64 configuration in Visual Studio 2013. For 64-bit versions of Windows, only the Visual Studio 2013 versions are available.

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

10.2 Building the Samples on a UNIX System

See the following sections for specific information about building the sample applications on your flavor of UNIX:

10.3 An Overview of the Sample Applications

This section describes the following sample applications.

> **Note:**
>
> Please note that not all of the sample applications are provided for both the Windows and UNIX platforms. See the heading of each application's subsection for clarification.

10.3.1 batch_process_ca

batch_process_ca demonstrates running Content Access in a separate process on multiple input files. It also allows the timing of each run.

The application is executed from the command line and takes several possible parameters:

```
batch_process_ca -f inputfile -o outputfile or [-d inputdir -o outputdir]
[-i iterations] [-q[2]] [-b]
```

- -f specifies the name of a single input file.
- -d specifies the name of an input directory of files.
- -o specifies the name of an output file if -f is being used, or the name of an output directory if -d is being used.
- -i is an optional parameter specifying the number of iterations to perform.
- -q and -q2 diminish the output to the screen.
- -b increases the amount of content in the output including processing tags and sub-documents.

10.3.2 casample

An example of a typical usage of the Outside In Content Access API is casample. Because this is intended as a simple template or reference for common Content Access usage, it creates only rudimentary output. However, it does initialize, exercise and cleanup Content Access output. Content Access requires the usage of the Outside In Data Access module. Therefore, this application also demonstrates usage of a portion of Data Access.

The application is executed from the command line and has one required parameter, the name of the input file. It will optionally take two other parameters: '-u' and an output file name.

```
casample input_file [-u outputfile]
```

10.3.3 extract_archive

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

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

```
extract_archive input_file output_directory
```

10.3.4 extract_object

extract_object demonstrates using Content Access to parse an input file and then using the DAObject API to extract all embedded objects.

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

```
extract_object input_file output_directory
```

10.3.5 memoryio

memoryio demonstrates how to use the redirected I/O and Content Access APIs to process an in-memory file.

The application is executed from the command line and takes only one parameter, the name of the input file:

```
memoryio input_file
```

10.3.6 parsepst

parsepst demonstrates how to parse email messages from a PST file using the CA API. It searches for messages received between two hard coded dates.

The application is executed from the command line and takes only one parameter, the name of the input file:

```
parsepst input_file
```

10.3.7 tademo (Windows Only)

The tademo sample application included with this product provides a simple demonstration of text access. The text from a file is read a block at a time and displayed in the tademo window. The TAReadFirst and TAReadNext functions are directly tied to menu options, and the block size may be set by the user. An option is also provided to save the text to a file.

10.3.8 taredir (UNIX Only)

This sample provides a means of using the API presented in this guide without the need for Motif libraries. All extracted text is output to the standard output device, or can be redirected to a file or another device.

The application is executed from the command line and takes only one parameter, the name of the input file:

```
taredir input_file
```

10.3.9 textdemo (UNIX Only)

The sample code in the textdemo files shows how to use the API presented in this guide. This application is essentially identical to the Windows-only application tademo, which is discussed at length in tademo (Windows Only).

A
Content Access Options

Options are parameters affecting the behavior of the Outside In Technology. These options are available to the developer when using Content Access. They are set using the DASetOption call. It is recommended that developers familiarize themselves with all of the options available.

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

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

The following types of options are covered:

A.1 Character Mapping

This section discusses character mapping.

A.1.1 SCCOPT_DEFAULTINPUTCHARSET

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

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

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

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Default

- CS_SYSTEMDEFAULT: Query the operating system.

Data

The data types are listed in charsets.h.

A.1.2 SCCOPT_OUTPUTCHARACTERSET

Any text returned by Content Access or Text Access will be in the specified character set.

Handle Types

VTHDOC, VTHCONTENT, VTHTEXT

Scope

Local

Data Type

VTDWORD

Default

If the option is not set, Content Access will use SO_ANSI1252 on all non-Windows platforms. The current ANSI code page will be retrieved on Windows using GetACP() with the result being mapped to match an Outside In Technology character set.

Data

One of the following values:

Value	Description
CS_DOS_437	U.S.
CS_DOS_737	Greek
CS_DOS_850	Latin-1
CS_DOS_852	Latin-2
CS_DOS_855	Cyrillic
CS_DOS_857	Turkish

Value	Description
CS_DOS_860	Portuguese
CS_DOS_863	French Canada
CS_DOS_865	Denmark, Norway-DAT
CS_DOS_866	Cyrillic
CS_DOS_869	Greece
CS_WINDOWS_874	Thailand
CS_WINDOWS_932	Japanese
CS_WINDOWS_936	Chinese GB
CS_WINDOWS_949	Korea (Wansung)
CS_WINDOWS_950	Hong Kong, Taiwan
CS_WINDOWS_1250	Windows Latin 2 (Central Europe)
CS_WINDOWS_1251	Windows Cyrillic (Slavic)
CS_WINDOWS_1252	Windows Latin 1 (ANSI)
CS_WINDOWS_1253	Windows Greek
CS_WINDOWS_1254	Windows Latin 5 (Turkish)
CS_WINDOWS_1255	Windows Hebrew
CS_WINDOWS_1256	Windows Arabic
CS_WINDOWS_1257	Windows Baltic
CS_UNICODE	Unicode
CS_ISO8859_1	Latin-1 - this is a subset of Windows 1252
CS_ISO8859_2	Latin-2
CS_ISO8859_3	Latin-3
CS_ISO8859_4	Latin-4
CS_ISO8859_5	Cyrillic
CS_ISO8859_6	Arabic
CS_ISO8859_7	Greek
CS_ISO8859_8	Hebrew
CS_ISO8859_9	Turkish

A.1.3 SCCOPT_UNMAPPABLECHAR

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTWORD

Data

The Unicode value for the character to use.

Default

- 0x002a = "*"

A.2 Input Handling

This section discusses input handling.

A.2.1 SCCOPT_EXTRACTXMPMETADATA

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

- SCCEX_IND_SUPPRESSPROPERTIES will not affect XMP, so if you turn XMP on, but also set SuppressProperties, you will still get the XMP.
- SCCEX_METADATAONLY will not guarantee that XMP is produced.

Handle Types

VTHDOC

Scope

Local (was Global prior to release 8.2.2)

Data Type

VTBOOL

Data

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

Default

- FALSE

A.2.2 SCCOPT_FALLBACKFORMAT

This option controls how files are handled when their specific application type cannot be determined. This normally affects all plain-text files, because plain-text files are

generally identified by process of elimination, for example, when a file isn't identified as having been created by a known application, it is treated as a plain-text file.

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

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Data

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

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

Default

- FI_TEXT

A.2.3 SCCOPT_FIFLAGS

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

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

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Data

One of the following values:

- SCCUT_FI_NORMAL: This is the default value. When this is set, standard file identification behavior occurs.
- SCCUT_FI_EXTENDEDTEST: If set, the File Identification code will run an extended test on all files that are not identified.

Default

- SCCUT_FI_NORMAL

A.2.4 SCCOPT_SYSTEMFLAGS

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

Handle Type

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

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

Default

0

A.2.5 SCCOPT_IGNORE_PASSWORD

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

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

Scope

Global

Data Type

VTBOOL

Data

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

Default

FALSE

A.2.6 SCCOPT_LOTUSNOTESDIRECTORY

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

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

Handle Types

NULL

Scope

Global

Data Type

VTLPBYTE

Data

A path to the Lotus Notes directory.

Default

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

A.2.7 SCCOPT_PARSEXMPMETADATA

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTBOOL

Data

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

Default

FALSE

A.2.8 SCCOPT_PDF_FILTER_REORDER_BIDI

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

Handle Types

VTHDOC, NULL

Scope

Global

Data Type

VTDWORD

Data

- SCCUT_FILTER_STANDARD_BIDI
- SCCUT_FILTER_REORDERED_BIDI

Default

SCCUT_FILTER_STANDARD_BIDI

A.2.9 SCCOPT_PROCESS_OLE_EMBEDDINGS

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

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

Handle Types

VTHDOC, NULL

Scope

Global

Data Type

VTWORD

Data

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

Default

SCCOPT_PROCESS_OLEEMBED_STANDARD

A.2.10 SCCOPT_TIMEZONE

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

> **Note:**
>
> Daylight savings is not supported. The sent time in msg files when viewed in Outlook can be an hour different from the time sent when an image of the msg file is created.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTLONG

Data

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

Default

- 0: GMT time

A.2.11 SCCOPT_HTML_COND_COMMENT_MODE

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

Handle Type

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

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

Default

HTML_COND_COMMENT_NONE

A.2.12 SCCOPT_PDF_FILTER_DROPHYPHENS

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

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

Handle Types

VTHDOC

Scope

Global

Data Type

VTBOOL

Data

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

Default

FALSE

A.2.13 SCCOPT_ARCFULLPATH

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

Data Type

VTBOOL

Data

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

Default

FALSE

A.2.14 SCCOPT_NULLREPLACECHAR

This option specifies a two-byte Unicode character that will be used to replace null characters if null path separators are being used. This option defaults to '/' and is valid for SearchML 3.x, SearchHTML, SearchText, Content Access and the DA APIs.

> **Note:**
>
> This is identical to SCCOPT_XML_NULLREPLACECHAR.

Handle Types

VTHDOC

Scope

Local

Data Type

VTWORD

Data

A two-byte Unicode character that will be used to replace null characters if null path separators are being used.

Default

0x002f = "/"

A.2.15 SCCOPT_EX_PERFORMANCEMODE

When possible, skip the processing of some or all style information. This should result in better performance, but certain output will no longer be available.

- SCCEX_PERFORMANCE_TEXTONLY - When this flag is set, no style information is processed in optimized filters. The following output won't be available even if they have been requested: character attributes, paragraph attributes, font names, and PDF Map Problem warnings. Not all input filters are optimized to work with this performance mode, but Microsoft Office, PDF, RTF, MSG, Mime, and HTML are included in the optimized list. If this flag is set and an input document for a non-optimized filter is encountered, this option will default back to SCCEX_PERFORMANCE_TEXTANDFONTS. Characters in symbol fonts use the font name as part of the character mapping process. Since the font name is not tracked, there may be minor mapping differences in these characters, but character counts should still be accurate.

- SCCEX_PERFORMANCE_TEXTANDFONTS - When this flag is set, minimal style information is tracked including character sets and font names. That information corrects the mapping differences in symbol characters, but doesn't give as much performance benefit as SCCEX_PERFORMANCE_TEXTONLY. This flag also works with all input filters.

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

One of the following:

- SCCEX_PERFORMANCE_NORMAL - Process the style information normally.
- SCCEX_PERFORMANCE_TEXTANDFONTS - Process only the font and character set information within a style.
- SCCEX_PERFORMANCE_TEXTONLY - Skip processing all style information.

Default

SCCEX_PERFORMANCE_NORMAL

> **Note:**
>
> This option is only supported in Search Export and Content Access. Attempting to use it with other products will lead to unpredictable results.

A.2.16 SCCOPT_GENERATEEXCELREVISIONS

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

Handle Types

VTHDOC

Scope

Global

Data Type

VTBOOL

Data

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

Default

FALSE

A.2.17 SCCOPT_PDF_FILTER_MAX_EMBEDDED_OBJECTS

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

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

Default

0

A.2.18 SCCOPT_PDF_FILTER_MAX_VECTOR_PATHS

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

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

Default

0

A.2.19 SCCOPT_PDF_FILTER_WORD_DELIM_FRACTION

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

Handle Types

NULL, VTHDOC

Scope

Local

Data Type

VTFLOAT

Data

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

Default

0.85

A.3 Compression

This section discusses compression.

A.3.1 SCCOPT_FILTERJPG

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

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

- JPG files

- Postscript files containing JPG images
- PDFs containing JPEG images

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

Handle Types

VTHDOC, HEXPORT

Scope

Global

Data Type

VTDWORD

Data

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

Default

SCCVW_FILTER_JPG_ENABLED

A.3.2 SCCOPT_FILTERLZW

This option can disable access to any files using Lempel-Ziv-Welch (LZW) compression, such as .GIF files, .ZIP files or self-extracting archive (.EXE) files containing "shrunk" files. Attempts to read such files when this option is enabled will fail and return the error SCCERR_UNSUPPORTEDCOMPRESSION. Unlike many other options, this option must be set programmatically, as it is not stored or read on startup.

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

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

Although this option can disable access to files in ZIP or EXE archives stored using LZW compression, any files in such archives that were stored using any other form of compression will still be accessible.

Handle Types

VTHDOC

Scope

Global

Data

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

Default

SCCVW_FILTER_LZW_ENABLED

A.4 Content Access Flags

The following section discusses content access flags.

A.4.1 SCCOPT_ENABLEALLSUBOBJECTS

Outside In has an internal flag that is used to optimize several of the input filters for searching. One of the side effects of this optimization is that many embedded bitmaps, including Progressive JPEG, aren't output by the filter. SCCOPT_ENABLEALLSUBOBJECTS can override this internal optimization.

Handle Types

VTHDOC

Scope

Global

Data Type

VTDWORD

Data

One of the following values:

- SCCUT_FILTER_ENABLEALLSUBOBJECTS: Override the optimizations.
- SCCUT_FILTER_NORMALSUBOBJECTS: Allow the optimizations.

Default

SCCUT_FILTER_NORMALSUBOBJECTS

A.4.2 SCCOPT_CA_FLAGS

This option allows the developer to set a flag to enable an option unique to Content Access.

Handle Types

VTHDOC

Scope

Local

Data Type

DWORD

Data

- SCCEX_IND_GENERATED: Includes data not originally stored as text in the input document. This can be important content the user would see when viewing the document in the original application (time and size information in archives, numbers in spreadsheets/databases, and so forth).
- SCCEX_IND_GENERATESYSTEMMETADATA: When this flag is set, system metadata will be generated. This text is "generated," so it will be affected by SCCEX_IND_GENERATED. This information is gathered through system calls and may adversely affect performance.

Default

- 0: The flag is turned off.

A.4.3 SCCOPT_FORMATFLAGS

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

Handle Types

VTHDOC

Scope

Local

Data Type

VTDWORD

Data

- SCCOPT_FLAGS_ISODATETIMES: When this flag is set, all Date and Time values are converted to the ISO 8601 standard. This conversion can only be performed using dates that are stored as numeric data within the original file.
- SCCOPT_FLAGS_STRICTFILEACCESS: When an embedded file or URL can't be opened with the full path, OIT will sometimes try and open the referenced file from other locations, including the current directory. When this flag is set, it will prevent OIT from trying to open the file from any location other than the fully qualified path or URL.

Default

0: All flags turned off

A.5 File System

This section discusses file systems.

A.5.1 SCCOPT_IO_BUFFERSIZE

This provides three options that allow the user to adjust buffer sizes to take advantage of faster computers/more memory. This is an advanced option that casual users of Content Access may ignore. This option allows the users to tune Content Access memory usage to a particular target machine. The number specified will be in kilobytes.

Handle Type

NULL, VTHDOC

Scope

Global

Data Type

SCCBUFFEROPTIONS Structure

Data

A buffer options structure

A.5.1.1 SCCBUFFEROPTIONS Structure

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

Parameters

- dwReadBufferSize: Used to define the number of bytes that will read from disk into memory at any given time. Once the buffer has data, further file reads will proceed within the buffer until the end of the buffer is reached, at which point the buffer will again be filled from the disk. This can lead to performance improvements in many file formats, regardless of the size of the document.
- dwMMapBufferSize: Used to define a maximum size that a document can be and use a memory-mapped I/O model. In this situation, the entire file is read from disk into memory and all further I/O is performed on the data in memory. This can lead to significantly improved performance, but note that either the entire file can be read into memory, or it cannot. If both of these buffers are set, then if the file is smaller than the dwMMapBufferSize, the entire file will be read into memory; if not, it will be read in blocks defined by the dwReadBufferSize.

- dwTempBufferSize: The maximum size that a temporary file can occupy in memory before being written to disk as a physical file. Storing temporary files in memory can boost performance on archives, files that have embedded objects or attachments. If set to 0, all temporary files will be written to disk.
- dwFlags
  - SCCBUFOPT_SET_READBUFSIZE 1
  - SCCBUFOPT_SET_MMAPBUFSIZE 2
  - SCCBUFOPT_SET_TEMPBUFSIZE 4

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

Default

The default settings for these options are:

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

Minimum and maximum sizes for each are:

- SCCBUFOPT_MIN_READBUFSIZE 1: Read one Kbyte at a time.
- SCCBUFOPT_MIN_MMAPBUFSIZE 0: Don't use memory-mapped input.
- SCCBUFOPT_MIN_TEMPBUFSIZE 0: Don't use memory temp files
- SCCBUFOPT_MAX_READBUFSIZE 0x003fffff, SCCBUFOPT_MAX_MMAPBUFSIZE 0x003fffff, SCCBUFOPT_MAX_TEMPBUFSIZE 0x003fffff: These maximums correspond to the largest file size possible under the 4GB DWORD limit.

A.5.2 SCCOPT_TEMPDIR

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

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

> **Note:**
>
> This option will be ignored if SCCOPT_REDIRECTTEMPFILE is set.

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

SCCUTTEMPDIRSPEC structure

A.5.2.1 SCCUTTEMPDIRSPEC Structure

This structure is used in the SCCOPT_TEMPDIR option.

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

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

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

- IOTYPE_ANSIPATH: Windows only. szTempDirName points to a NULL-terminated full path name using the ANSI character set and FAT 8.3 (Win16) or NTFS (Win32 and Win64) file name conventions.
- IOTYPE_UNICODEPATH: Windows only. szTempDirName points to a NULL-terminated full path name using the Unicode character set and NTFS file name conventions. Note that the length of the path name is limited to SCCUT_FILENAMEMAX bytes, or (SCCUT_FILENAMEMAX / 2) double-byte Unicode characters.
- IOTYPE_UNIXPATH: X Windows on UNIX platforms only. szTempDirName points to a NULL-terminated full path name using the system default character set and UNIX path conventions.

Specifically not supported at this time is IOTYPE_REDIRECT.

Parameters

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

A.5.3 SCCOPT_DOCUMENTMEMORYMODE

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

Handle Types

NULL, VTHDOC

Scope

Global

Data Type

VTDWORD

Parameters

- SCCDOCUMENTMEMORYMODE_SMALLEST (4MB)
- SCCDOCUMENTMEMORYMODE_SMALL 2 (16MB)
- SCCDOCUMENTMEMORYMODE_MEDIUM 3 (64MB)
- SCCDOCUMENTMEMORYMODE_LARGE (256MB)
- SCCDOCUMENTMEMORYMODE_LARGEST (1 GB)

Default

SCCDOCUMENTMEMORYMODE_LARGE (256MB)

A.5.4 SCCOPT_REDIRECTTEMPFILE

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

Handle Types

NULL, VTHDOC

Scope

Global (not persistent)

Data Type

VTLPVOID: pCallbackFunc

Function pointer of the redirect IO callback.

Redirect call back function:

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

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

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W