Connector Framework Server

Software Version: 11.6

Administration Guide

Document Release Date: February 2018
Software Release Date: February 2018

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To verify you are using the most recent edition of a document, go to https://softwaresupport.softwaregrp.com/group/softwaresupport/search-result?doctype=online help.

This site requires you to sign in with a Software Passport. You can register for a Passport through a link on the site.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your Micro Focus sales representative for details.

Support

Visit the Micro Focus Software Support Online website at https://softwaresupport.softwaregrp.com.

This website provides contact information and details about the products, services, and support that Micro Focus offers.

Micro Focus online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Access the Software Licenses and Downloads portal
- Download software patches
- Access product documentation
- Manage support contracts

- Look up Micro Focus support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require you to register as a Passport user and sign in. Many also require a support contract.

You can register for a Software Passport through a link on the Software Support Online site.

To find more information about access levels, go to https://softwaresupport.softwaregrp.com/web/softwaresupport/access-levels.

About this PDF version of online Help

This document is a PDF version of the online Help.

This PDF file is provided so you can easily print multiple topics or read the online Help.

Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online Help.

Contents

Chapter 1: Introduction

This section provides an overview of Connector Framework Server.

Connector Framework Server

Connector Framework Server (CFS) processes the information that is retrieved by connectors, and then indexes the information into one or more indexes, such as IDOL Server or Haven OnDemand.

Connectors send information to CFS in the form of documents. A *document* is a collection of metadata and, usually, an associated source file. The metadata describes the location of the file or record that was retrieved, and other information that was extracted by the connector. For example, a document sent for ingestion by a Web Connector includes the URL of the page and the links that were extracted from the page when it was crawled. The Web Connector provides the downloaded HTML in an associated file so that it can be processed by CFS.

Sometimes a document does not have an associated source file. For example, if you retrieve information from a database using the ODBC Connector, the documents sent for ingestion contain the information extracted by your chosen query, and might not have an associated file. These documents are referred to as having *metadata only*.

CFS uses KeyView to extract information from the source file. Some source files are container files, such as zip archives, and these are extracted. CFS then uses KeyView to obtain text and file-specific metadata from the file, and adds it to the document. The original source file is discarded before the document is indexed. This allows IDOL to search and categorize documents, and perform other operations, without needing to process the information from a repository in its native format.

CFS provides features to manipulate and enrich documents. For example, you can send media files to an IDOL Media Server and perform tasks such as optical character recognition and face recognition. This adds additional information to the IDOL document, so that when a user queries IDOL the results include relevant images, audio, and video files. CFS also supports the Lua scripting language so that you can write your own tasks and develop custom processing rules.

A single CFS can process information from any number of connectors. For example, a CFS might process files retrieved by a File System Connector, web pages retrieved by a Web Connector, and e-mail messages retrieved by an Exchange Connector. Alternatively, you or an application can send information to CFS directly.

Filter Documents and Extract Subfiles

CFS uses KeyView to extract meaningful information from the files retrieved by a connector. KeyView can extract the file content, metadata, and subfiles from over 1,000 different file types.

- *File content* is the main content of a file, for example the body of an e-mail message.
- *Metadata* is information about a file itself, for example the sender of an e-mail message or the date and time when it was received.
- *Subfiles* are files that are contained within the main file. For example, an e-mail message might contain embedded images or attachments that you want to index.

Manipulate and Enrich Documents

CFS provides features to manipulate and enrich documents. *Enriching* a document means adding additional information, or improving the quality and usefulness of the information, before the document is indexed into IDOL. For example, you can:

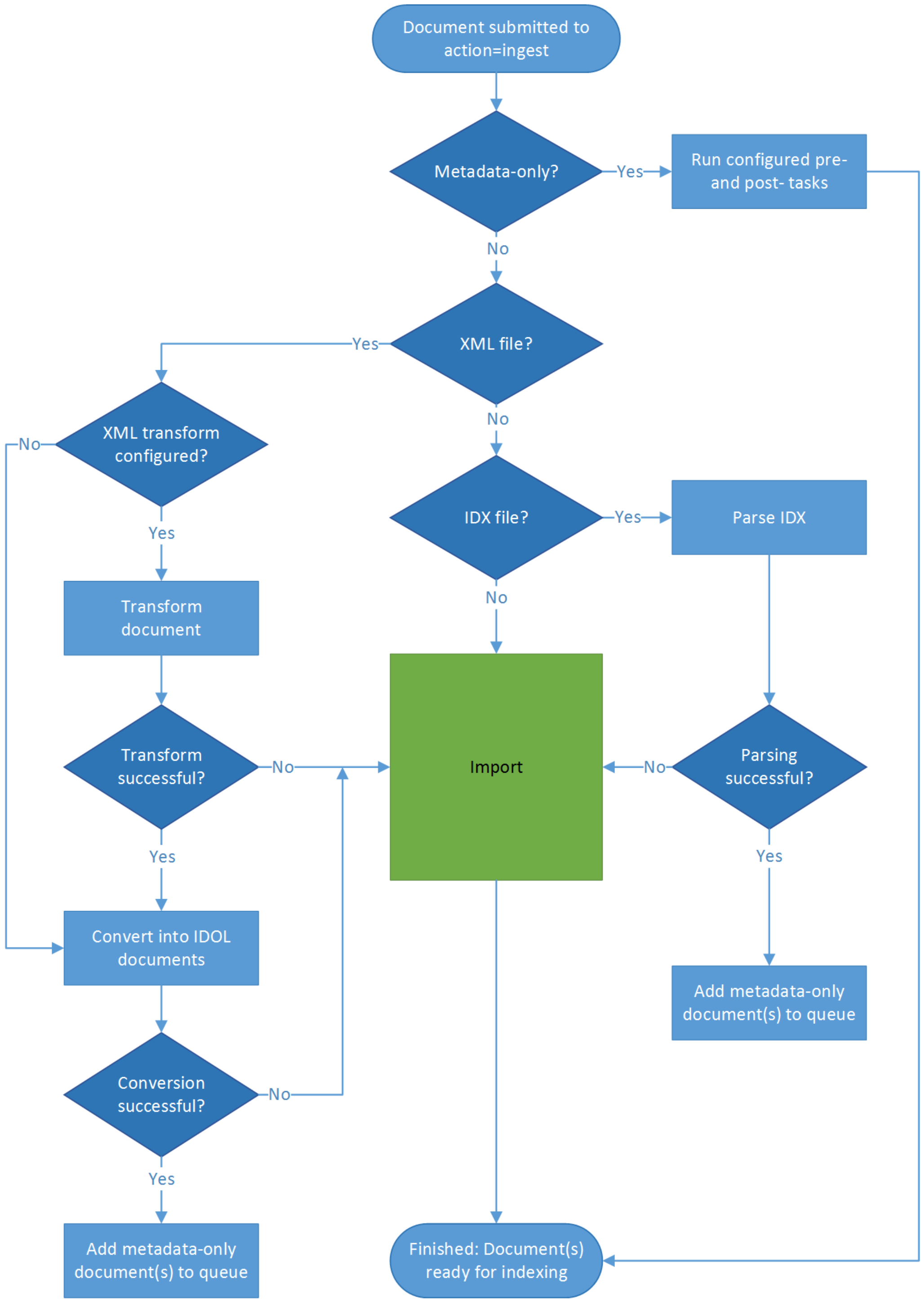
- Add additional fields to a document.
- Extract content from HTML pages, discarding irrelevant content such as headers, sidebars, advertisements, and scripts.
- Split long documents into multiple sections. This can improve performance when you query IDOL, because IDOL can return a specific part of a document in response to a query.
- Standardize field names, so that documents that originated from different repositories use the same fields to store the same type of information.
- Perform *Eduction* on document fields. Eduction extracts *entities* from a document, and writes them to specific document fields. An entity can be a word, phrase, or block of information - for example an address or telephone number.
- Perform analysis on image and video files and add the results to the document. Examples of media analysis include optical character recognition (OCR), face detection and recognition, and object recognition. To analyze media you must have an IDOL Media Server.
- Extract speech from audio and video files, and add the transcription to the document content. To analyze speech you must have an IDOL Speech Server.
- Reject documents that do not contain content in a specific language.

The simplest way to manipulate documents is to use the *import tasks* that are included with CFS. For information about the tasks that are available, see Manipulate and Enrich Documents, on page 47. You can configure these tasks by modifying configuration parameters in the CFS configuration file.

CFS also supports Lua, an embedded scripting language. You can write Lua scripts to manipulate documents and define custom processing rules. For information about the Lua functions that are provided with CFS, refer to the *Connector Framework Server Reference*.

The Ingestion Process

The following chart provides a summary of the ingestion process.


Document submitted to action=ingest
Metadata-only?
Yes
Run configured pre- and post- tasks
No
XML file?
Yes
No
XML transform configured?
No
Yes
IDX file?
Yes
Parse IDX
No
Transform document
Transform successful?
No
Import
No
Parsing successful?
Yes
Yes
Convert into IDOL documents
Add metadata-only document(s) to queue
Conversion successful?
No
Yes
Add metadata-only document(s) to queue
Finished: Document(s) ready for indexing

Documents are submitted to Connector Framework Server through the `ingest` action. If the document has metadata only, CFS runs any processing tasks that have been configured and the document is then ready for indexing. If the document has an associated file then the ingestion process depends on the file format.

- **All files apart from IDOL IDX and XML**. Most documents that have an associated file are added to the import queue so that the information in the file can be extracted by KeyView or other processing tasks. For information about the import process, see The Import Process, on the next page.
- **IDOL IDX files**. An IDX file contains one or more documents in IDOL IDX format, so CFS attempts to parse the file. If parsing is successful then the IDOL documents are returned to the ingest queue as metadata-only documents. If parsing is not successful then CFS adds the document to the import queue so that the IDX file is processed by KeyView. Parsing an IDX file is preferable to processing it with KeyView, because although KeyView can extract the text, it cannot extract the structure information that divides the text into separate documents, content sections, and metadata fields.
- **XML files**. Many systems export information in XML format and CFS has features to help you convert XML into IDOL documents.

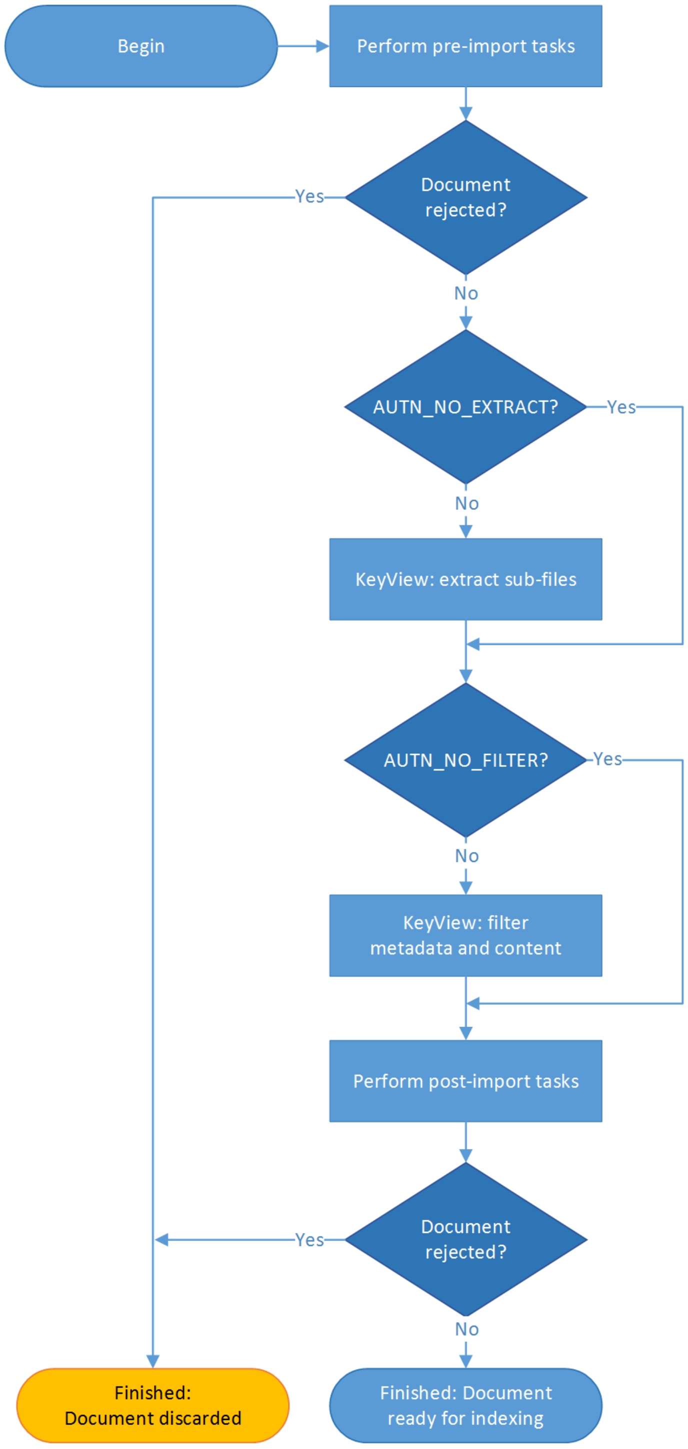
  CFS can run a transformation on an ingested XML file. This is an optional step but can be useful in cases where your XML files do not resemble IDOL documents or you are processing XML from many sources and the files have different schemas. You can configure any number of transformations and CFS runs the first transformation where the ingested XML matches the specified schema. You can also configure a default transformation that CFS runs when an XML file does not match any of your schemas. When a transformation is configured but is not successful, CFS adds the document to the import queue so that the XML is processed by KeyView.

  After an XML transformation is successful or when transformation is not configured, CFS attempts to convert the XML into IDOL documents. The conversion is performed by mapping elements in the XML to IDOL documents and document fields. If the conversion is successful the resulting documents are returned to the ingest queue as metadata-only documents. If the conversion does not result in any IDOL documents but the XML was transformed after matching a schema, CFS does not consider this as a failure and does not index any documents. Otherwise, CFS adds the document to the import queue so that the XML is processed by KeyView.

  Parsing an XML file is usually preferable to processing it with KeyView, because although KeyView can extract the text it does not preserve the structure information (the XML tags are discarded).

The Import Process

The following chart provides a summary of the import process.


Begin
Perform pre-import tasks
Document rejected?
Yes
No
AUTN_NO_EXTRACT?
Yes
No
KeyView: extract sub-files
AUTN_NO_FILTER?
Yes
No
KeyView: filter metadata and content
Perform post-import tasks
Document rejected?
Yes
No
Finished: Document discarded
Finished: Document ready for indexing

1. CFS takes a document from the import queue.
2. CFS performs the pre-import tasks that are configured in its configuration file. Pre-import tasks occur before files are processed by KeyView. You can use pre-import tasks to manipulate and enrich documents (see Manipulate and Enrich Documents, on page 10). Sometimes it is important to run tasks before KeyView processing. For example, if you send an audio file to Media Server for analysis, you might not want to process it with KeyView.

   > **TIP:**
   > Both pre- and post-import tasks can reject a document, so that it is discarded and not indexed. You might configure CFS to reject a document if the associated file does not contain useful content. Documents are not rejected when an import task fails - in that case CFS continues processing the document.

3. Unless the document contains the metadata field `AUTN_NO_EXTRACT`, CFS uses KeyView to extract sub-files. Examples of files that have sub-files include e-mail messages (which have attachments) and zip files (which contain other files). CFS creates a new document for each sub-file and adds the new documents to the import queue to be processed separately.
4. Unless the document contains the metadata field `AUTN_NO_FILTER`, CFS uses KeyView to filter the associated source file. Filtering extracts the text from a file. An office document is likely to contain useful text, while an archive file (for example a zip file) or a media file is unlikely to have textual content.

   > **TIP:**
   > Although media files (images, audio, and video) do not contain text, you can extract useful information by sending the files to an IDOL Media Server.

5. CFS performs the post-import tasks that are configured in its configuration file.
6. Processing is complete and the document is ready to be indexed.

Index Documents

After CFS finishes processing documents, it automatically indexes them into one or more indexes. You can index documents into:

- **IDOL Server** (or send them to a *Distributed Index Handler*, so that they can be distributed across multiple IDOL servers).
- **Haven OnDemand**.
- **Vertica**.

The IDOL Platform

At the core of Connector Framework Server is the *Intelligent Data Operating Layer* (IDOL).

IDOL gathers and processes unstructured, semi-structured, and structured information in any format from multiple repositories using IDOL connectors and a global relational index. It can automatically form a contextual understanding of the information in real time, linking disparate data sources together based on the concepts contained within them. For example, IDOL can automatically link concepts contained in an email message to a recorded phone conversation, that can be associated with a stock trade. This information is then imported into a format that is easily searchable, adding advanced retrieval, collaboration, and personalization to an application that integrates the technology.

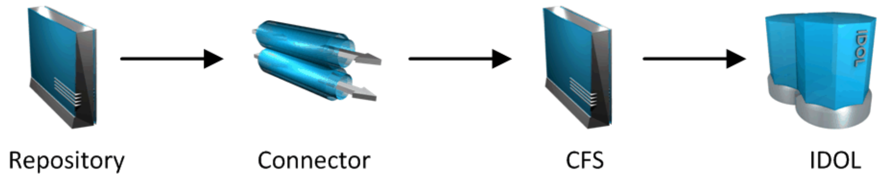
For more information on IDOL, see the *IDOL Getting Started Guide*.

System Architecture

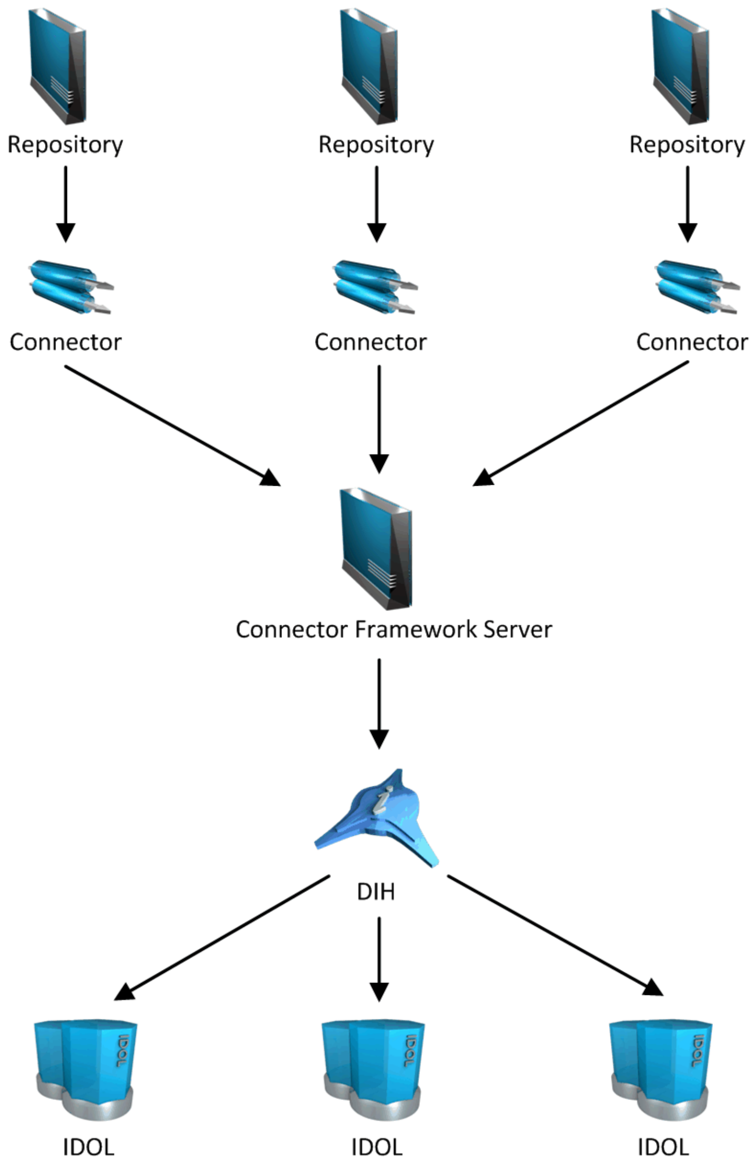
An IDOL infrastructure can include the following components:

- **Connectors**. Connectors extract data from repositories and send the data to CFS.
- **Connector Framework Server**.
- **IDOL Server**. IDOL Server provides features to analyze unstructured information and extract meaning from that information.
- **Distributed Index Handler (DIH)**. The Distributed Index Handler distributes data across multiple IDOL servers. Using multiple IDOL servers can increase the availability and scalability of the system.

These components can be installed in many different configurations. The simplest installation consists of a single connector, a single CFS, and a single IDOL server.



A more complex configuration might include more than one connector, or use a Distributed Index Handler (DIH) to index content across multiple IDOL servers.



OEM Certification

Connector Framework Server works in OEM licensed environments.

Related Documentation

The following documents provide more details on Connector Framework Server.

- *Connector Framework Server Reference*

  The *Connector Framework Server Reference* describes the configuration parameters and actions that are supported by CFS.
- *IDOL Server Administration Guide*

  The *IDOL Server Administration Guide* describes the operations that IDOL Server can perform, and describes how to set them up.
- *Distributed Index Handler (DIH) Administration Guide*

  This guide describes how you can use a DIH to distribute aggregated documents across multiple IDOL Servers.
- *License Server Administration Guide*

  This guide describes how to use a License Server to license multiple IDOL services.

Display Online Help

You can display the Connector Framework Server Reference by sending an action from your web browser. The Connector Framework Server Reference describes the actions and configuration parameters that you can use with Connector Framework Server.

For Connector Framework Server to display help, the help data file (`help.dat`) must be available in the installation folder.

To display help for Connector Framework Server

1. Start Connector Framework Server.
2. Send the following action from your web browser:

   ```
   http://host:port/action=Help
   ```

   where:

host	is the IP address or name of the machine on which Connector Framework Server is installed.
port	is the ACI port by which you send actions to Connector Framework Server (set by the `Port` parameter in the `[Server]` section of the configuration file).

   For example:

   ```
   http://12.3.4.56:9000/action=help
   ```

Chapter 2: Configure Connector Framework Server

This section describes how to configure CFS.

Connector Framework Server Configuration File

To configure CFS, modify the configuration file. The file is located in the CFS installation folder and can be modified with a text editor.

The parameters in the configuration file are divided into sections that represent CFS functionality. CFS supports standard Server, Service, Logging, and License parameters.

Service Section

The `[Service]` section specifies the service port used by CFS.

Server Section

The `[Server]` section specifies the ACI port of the Connector Framework Server. When you configure connectors, the `IngestPort` parameter in the connector configuration file should point to this port.

Actions Section

The `[Actions]` section specifies how CFS processes actions that are sent to the ACI port.

Logging Section

The `[Logging]` section contains configuration parameters that determine how messages are logged. You can create separate log streams for different message types. The configuration file also contains a section to configure each of the log streams.

Indexing Section

The `[Indexing]` section specifies the host name or IP address, and port, of machines where data is sent after it has been processed by CFS. This is usually the IP address and ACI port of an IDOL Server. You can use other indexing parameters to specify how data is indexed.

ImportService Section

The `[ImportService]` section specifies details for KeyView.

ImportTasks Section

The `[ImportTasks]` section is used to set up custom import tasks. CFS performs these tasks on data before it is indexed into IDOL Server. For more information about Import Tasks, see Manipulate and Enrich Documents, on page 47.

IndexTasks Section

The `[IndexTasks]` section is used to set up custom index tasks. IDOL connectors detect when documents are updated or removed from a repository. The connectors pass this information to CFS so that the documents can be updated or removed from IDOL Server. When CFS receives this information, it can perform custom Index tasks before the information is sent to IDOL. For more information about Index tasks, see Manipulate and Enrich Documents, on page 47.

Related Topics

- Example Configuration File, on page 27
- Customize Logging, on page 113

Modify Configuration Parameter Values

You modify Connector Framework Server configuration parameters by directly editing the parameters in the configuration file. When you set configuration parameter values, you must use UTF-8.

CAUTION:
You must stop and restart Connector Framework Server for new configuration settings to take effect.

This section describes how to enter parameter values in the configuration file.

Enter Boolean Values

The following settings for Boolean parameters are interchangeable:

```
TRUE = true = ON = on = Y = y = 1

FALSE = false = OFF = off = N = n = 0
```

Enter String Values

To enter a comma-separated list of strings when one of the strings contains a comma, you can indicate the start and the end of the string with quotation marks, for example:

`ParameterName``=`**`cat,dog,bird,"wing,beak",turtle`**

Alternatively, you can escape the comma with a backslash:

`ParameterName``=`**`cat,dog,bird,wing\,beak,turtle`**

If any string in a comma-separated list contains quotation marks, you must put this string into quotation marks and escape each quotation mark in the string by inserting a backslash before it. For example:

`ParameterName``=`**`"<font face=\"arial\" size=\"+1\"><b>","<p>"`**

Here, quotation marks indicate the beginning and end of the string. All quotation marks that are contained in the string are escaped.

Configure Connector Framework Server

This section describes how to configure CFS.

To configure CFS

1. Stop CFS, if it is running.
2. Open the CFS configuration file.
3. In the `[Service]` section, specify the service port:

`ServicePort`	The port for CFS to use as the service port.

4. In the `[Server]` section, set the ACI port:

`Port`	The port for CFS to use as the ACI port.

5. (Optional) In the `[ImportService]` section, you can set parameters to configure KeyView. You can choose the number of threads to use, specify the folders to use for extracting files, and customize how documents are imported.

`ThreadCount`	The number of threads to use for importing documents.

   For information about the configuration parameters that you can set, refer to the *Connector Framework Server Reference*.
6. Save the configuration file.

Related Topics

- Start and Stop Connector Framework Server, on page 29
- Connector Framework Server Configuration File, on page 18
- Customize Logging, on page 113

Include an External Configuration File

You can share configuration sections or parameters between ACI server configuration files. The following sections describe different ways to include content from an external configuration file.

You can include a configuration file in its entirety, specified configuration sections, or a single parameter.

When you include content from an external configuration file, the `GetConfig` and `ValidateConfig` actions operate on the combined configuration, after any external content is merged in.

In the procedures in the following sections, you can specify external configuration file locations by using absolute paths, relative paths, and network locations. For example:

```
../sharedconfig.cfg
K:\sharedconfig\sharedsettings.cfg
\\example.com\shared\idol.cfg
file://example.com/shared/idol.cfg
```

Relative paths are relative to the primary configuration file.

NOTE:
You can use nested inclusions, for example, you can refer to a shared configuration file that references a third file. However, the external configuration files must not refer back to your original configuration file. These circular references result in an error, and Connector Framework Server does not start.

Similarly, you cannot use any of these methods to refer to a different section in your primary configuration file.

Include the Whole External Configuration File

This method allows you to import the whole external configuration file at a specified point in your configuration file.

To include the whole external configuration file

1. Open your configuration file in a text editor.
2. Find the place in the configuration file where you want to add the external configuration file.
3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. For example:

   ```
   < "K:\sharedconfig\sharedsettings.cfg"
   ```

4. Save and close the configuration file.

Include Sections of an External Configuration File

This method allows you to import one or more configuration sections from an external configuration file at a specified point in your configuration file. You can include a whole configuration section in this way, but the configuration section name in the external file must exactly match what you want to use in your file. If you want to use a configuration section from the external file with a different name, see Merge a Section from an External Configuration File, on the next page.

To include sections of an external configuration file

1. Open your configuration file in a text editor.
2. Find the place in the configuration file where you want to add the external configuration file section.
3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. After the configuration file name, add the configuration section name that you want to include. For example:

   ```
   < "K:\sharedconfig\extrasettings.cfg" [License]
   ```

   > **NOTE:**
   > You cannot include a section that already exists in your configuration file.

4. Save and close the configuration file.

Include a Parameter from an External Configuration File

This method allows you to import a parameter from an external configuration file at a specified point in your configuration file. You can include a section or a single parameter in this way, but the value in the external file must exactly match what you want to use in your file.

To include a parameter from an external configuration file

1. Open your configuration file in a text editor.
2. Find the place in the configuration file where you want to add the parameter from the external configuration file.
3. On a new line, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. After the configuration file name, add the name of the configuration section name that contains the parameter, followed by the parameter name. For example:

   ```
   < "license.cfg" [License] LicenseServerHost
   ```

   To specify a default value for the parameter, in case it does not exist in the external configuration file, specify the configuration section, parameter name, and then an equals sign (=) followed by the default value. For example:

   ```
   < "license.cfg" [License] LicenseServerHost=localhost
   ```

4. Save and close the configuration file.

Merge a Section from an External Configuration File

This method allows you to include a configuration section from an external configuration file as part of your Connector Framework Server configuration file. For example, you might want to specify a standard SSL configuration section in an external file and share it between several servers. You can use this method if the configuration section that you want to import has a different name to the one you want to use.

To merge a configuration section from an external configuration file

1. Open your configuration file in a text editor.
2. Find or create the configuration section that you want to include from an external file. For example:

   ```
   [SSLOptions1]
   ```

3. After the configuration section name, type a left angle bracket (<), followed by the path to and name of the external configuration file, in quotation marks (""). You can use relative paths and network locations. For example:

   ```
   [SSLOptions1] < "../sharedconfig/ssloptions.cfg"
   ```

   If the configuration section name in the external configuration file does not match the name that you want to use in your configuration file, specify the section to import after the configuration file name. For example:

   ```
   [SSLOptions1] < "../sharedconfig/ssloptions.cfg" [SharedSSLOptions]
   ```

   In this example, Connector Framework Server uses the values in the `[SharedSSLOptions]` section of the external configuration file as the values in the `[SSLOptions1]` section of the Connector Framework Server configuration file.

   NOTE:
   You can include additional configuration parameters in the section in your file. If these parameters also exist in the imported external configuration file, Connector Framework Server uses the values in the local configuration file. For example:

   ```
   [SSLOptions1] < "ssloptions.cfg" [SharedSSLOptions]
   SSLCACertificatesPath=C:\IDOL\HTTPConnector\CACERTS\
   ```

4. Save and close the configuration file.

Encrypt Passwords

Micro Focus recommends that you encrypt all passwords that you enter into a configuration file.

Create a Key File

A key file is required to use AES encryption.

To create a new key file

1. Open a command-line window and change directory to the Connector Framework Server installation folder.
2. At the command line, type:

   ```
   autpassword -x -tAES -oKeyFile=./MyKeyFile.ky
   ```

   A new key file is created with the name `MyKeyFile.ky`

CAUTION:
To keep your passwords secure, you must protect the key file. Set the permissions on the key file so that only authorized users and processes can read it. Connector Framework Server must be able to read the key file to decrypt passwords, so do not move or rename it.

Encrypt a Password

The following procedure describes how to encrypt a password.

To encrypt a password

1. Open a command-line window and change directory to the Connector Framework Server installation folder.
2. At the command line, type:

   ```
   autpassword -e -tEncryptionType [-oKeyFile] [-cFILE -sSECTION -pPARAMETER]
   PasswordString
   ```

   where:

Option	Description
`-t` *`EncryptionType`*	The type of encryption to use: • **`Basic`** • **`AES`** For example: **`-tAES`** **NOTE:** AES is more secure than basic encryption.
`-o`*`KeyFile`*	AES encryption requires a key file. This option specifies the path and file name of a key file. The key file must contain 64 hexadecimal characters. For example: **`-oKeyFile=./key.ky`**
`-cFILE -sSECTION -pPARAMETER`	(Optional) You can use these options to write the password directly into a configuration file. You must specify all three options. • **`-c`**. The configuration file in which to write the encrypted password. • **`-s`**. The name of the section in the configuration file in which to write the password. • **`-p`**. The name of the parameter in which to write the encrypted

Option	Description
	password. For example: **`-c./Config.cfg -sMyTask -pPassword`**
`PasswordString`	The password to encrypt.

For example:

```
autpassword -e -tBASIC MyPassword
```

```
autpassword -e -tAES -oKeyFile=./key.ky MyPassword
```

```
autpassword -e -tAES -oKeyFile=./key.ky -c./Config.cfg -sDefault -pPassword
MyPassword
```

The password is returned, or written to the configuration file.

Decrypt a Password

The following procedure describes how to decrypt a password.

To decrypt a password

1. Open a command-line window and change directory to the Connector Framework Server installation folder.
2. At the command line, type:

   ```
   autpassword -d -tEncryptionType [-oKeyFile] PasswordString
   ```

   where:

Option	Description
`-t` *`EncryptionType`*	The type of encryption: • **`Basic`** • **`AES`** For example: **`-tAES`**
`-o`*`KeyFile`*	AES encryption and decryption requires a key file. This option specifies the path and file name of the key file used to decrypt the password. For example: **`-oKeyFile=./key.ky`**
`PasswordString`	The password to decrypt.

   For example:

   ```
   autpassword -d -tBASIC 9t3M3t7awt/J8A
   ```

   ```
   autpassword -d -tAES -oKeyFile=./key.ky 9t3M3t7awt/J8A
   ```

   The password is returned in plain text.

Configure Client Authorization

You can configure Connector Framework Server to authorize different operations for different connections.

Authorization roles define a set of operations for a set of users. You define the operations by using the `StandardRoles` configuration parameter, or by explicitly defining a list of allowed actions in the `Actions` and `ServiceActions` parameters. You define the authorized users by using a client IP address, SSL identities, and GSS principals, depending on your security and system configuration.

For more information about the available parameters, see the *Connector Framework Server Reference*.

To configure authorization roles

1. Open your configuration file in a text editor.
2. Find the `[AuthorizationRoles]` section, or create one if it does not exist.
3. In the `[AuthorizationRoles]` section, list the user authorization roles that you want to create. For example:

   ```
   [AuthorizationRoles]
   0=AdminRole
   1=UserRole
   ```

4. Create a section for each authorization role that you listed. The section name must match the name that you set in the `[AuthorizationRoles]` list. For example:

   ```
   [AdminRole]
   ```

5. In the section for each role, define the operations that you want the role to be able to perform. You can set `StandardRoles` to a list of appropriate values, or specify an explicit list of allowed actions by using `Actions`, and `ServiceActions`. For example:

   ```
   [AdminRole]
   StandardRoles=Admin,ServiceControl,ServiceStatus

   [UserRole]
   Actions=GetVersion
   ServiceActions=GetStatus
   ```

   > **NOTE:**
   > The standard roles do not overlap. If you want a particular role to be able to perform all actions, you must include all the standard roles, or ensure that the clients, SSL identities, and so on, are assigned to all relevant roles.

6. In the section for each role, define the access permissions for the role, by setting `Clients`, `SSLIdentities`, and `GSSPrincipals`, as appropriate. If an incoming connection matches one of the allowed clients, principals, or SSL identities, the user has permission to perform the operations allowed by the role. For example:

   ```
   [AdminRole]
   StandardRoles=Admin,ServiceControl,ServiceStatus
   ```

```
Clients=localhost
SSLIdentities=admin.example.com
```

7. Save and close the configuration file.
8. Restart Connector Framework Server for your changes to take effect.

Example Configuration File

This section contains an example configuration file, which meets the minimum configuration requirements.

```
[Service]
Port=17000

[Server]
Port=7000
MaxInputString=-1
MaxFileUploadSize=-1
XSLTemplates=TRUE

[AuthorizationRoles]
0=AdminRole
1=QueryRole

[AdminRole]
StandardRoles=admin,servicecontrol,query,servicestatus
Clients=::1,127.0.0.1

[QueryRole]
StandardRoles=query,servicestatus
Clients=*

[Actions]
MaxQueueSize=100

[Logging]
LogLevel=NORMAL
0=ApplicationLogStream
1=ActionLogStream
2=ImportLogStream
3=IndexLogStream

[ApplicationLogStream]
LogTypeCSVs=application
LogFile=application.log

[ActionLogStream]
LogTypeCSVs=action
```

```
LogFile=action.log

[ImportLogStream]
LogTypeCSVs=import
LogFile=import.log

[IndexLogStream]
LogTypeCSVs=indexer
LogFile=indexer.log

[Indexing]
IndexerSections=IdolServer
IndexBatchSize=1000
IndexTimeInterval=300

[IdolServer]
Host=idol
Port=9000
DefaultDatabaseName=News
SSLConfig=SSLOptions

[SSLOptions]
SSLMethod=SSLV23

[ImportService]
KeyviewDirectory=filters
ExtractDirectory=temp
ThreadCount=3
ImportInheritFieldsCSV=AUTN_GROUP,AUTN_IDENTIFIER,DREDBNAME

[ImportTasks]
//Post0=lua:<path_to_lua_file>
Post0=IdxWriter:C:\Autonomy\ConnectorFramework\IDX\output.idx
```

Chapter 3: Start and Stop Connector Framework Server

This section describes how to start and stop CFS.

Start Connector Framework Server

This section describes how to start Connector Framework Server.

To start CFS on Windows

1. Open the Windows Services dialog box.
2. Select the **ConnectorFramework** service (you might have chosen a different name for the service during the installation process).
3. Click **Start**.
4. (*Optional*). To verify that CFS is ready, send the following action to the ACI port.

   ```
   http://host:port/action=getstatus
   ```

   A response is displayed.

To start CFS on UNIX

1. Change to the CFS installation directory.
2. Run the start script by using the following command.

   ```
   ./startconnectorFramework.sh
   ```

Stop Connector Framework Server

This section describes how to stop Connector Framework Server.

To stop CFS on Windows

1. Open the Windows Services dialog box.
2. Select the **ConnectorFramework** service (you might have chosen a different name for the service during the installation process).
3. Click **Stop**, and close the Windows Services dialog box.

To stop CFS on UNIX

1. Change to the CFS installation directory.
2. Run the stop script by using the following command.

   ```
   ./stopconnectorFramework.sh
   ```

Chapter 4: Send Actions to Connector Framework Server

This section describes how to send actions to Connector Framework Server.

Send Actions to Connector Framework Server

Connector Framework Server actions are HTTP requests, which you can send, for example, from your web browser. The general syntax of these actions is:

`http://`*`host`*`:`*`port`*`/action=`*`action`*`&`*`parameters`*

where:

`host`	is the IP address or name of the machine where Connector Framework Server is installed.
`port`	is the Connector Framework Server ACI port. The ACI port is specified by the `Port` parameter in the `[Server]` section of the Connector Framework Server configuration file. For more information about the `Port` parameter, see the *Connector Framework Server Reference*.
`action`	is the name of the action you want to run.
`parameters`	are the required and optional parameters for the action.

NOTE:
Separate individual parameters with an ampersand (&). Separate parameter names from values with an equals sign (=). You must percent-encode all parameter values.

For more information about actions, see the *Connector Framework Server Reference*.

Asynchronous Actions

When you send an asynchronous action to Connector Framework Server, the CFS adds the task to a queue and returns a token. Connector Framework Server performs the task when a thread becomes available. You can use the token with the `QueueInfo` action to check the status of the action and retrieve the results of the action.

Most of the actions sent to CFS are `ingest` actions, so when you use the `QueueInfo` action, query the `ingest` action queue, for example:

```
/action=QueueInfo&QueueName=ingest&QueueAction=GetStatus
```

Check the Status of an Asynchronous Action

To check the status of an asynchronous action, use the token that was returned by Connector Framework Server with the `QueueInfo` action. For more information about the `QueueInfo` action, refer to the *Connector Framework Server Reference*.

To check the status of an asynchronous action

- Send the `QueueInfo` action to Connector Framework Server with the following parameters.

`QueueName`	The name of the action queue that you want to check.
`QueueAction`	The action to perform. Set this parameter to **`GetStatus`**.
`Token`	(Optional) The token that the asynchronous action returned. If you do not specify a token, Connector Framework Server returns the status of every action in the queue.

For example:

```
/action=QueueInfo&QueueName=ingest&QueueAction=getstatus&Token=...
```

Cancel an Asynchronous Action that is Queued

To cancel an asynchronous action that is waiting in a queue, use the following procedure.

To cancel an asynchronous action that is queued

- Send the `QueueInfo` action to Connector Framework Server with the following parameters.

`QueueName`	The name of the action queue that contains the action to cancel.
`QueueAction`	The action to perform . Set this parameter to **`Cancel`**.
`Token`	The token that the asynchronous action returned.

For example:

```
/action=QueueInfo&QueueName=ingest&QueueAction=Cancel&Token=...
```

Stop an Asynchronous Action that is Running

You can stop an asynchronous action at any point.

To stop an asynchronous action that is running

- Send the `QueueInfo` action to Connector Framework Server with the following parameters.

`QueueName`	The name of the action queue that contains the action to stop.
`QueueAction`	The action to perform. Set this parameter to **`Stop`**.
`Token`	The token that the asynchronous action returned.

For example:

```
/action=QueueInfo&QueueName=ingest&QueueAction=Stop&Token=...
```

Store Action Queues in an External Database

Connector Framework Server provides asynchronous actions. Each asynchronous action has a queue to store requests until threads become available to process them. You can configure Connector Framework Server to store these queues either in an internal database file, or in an external database hosted on a database server.

The default configuration stores queues in an internal database. Using this type of database does not require any additional configuration.

You might want to store the action queues in an external database so that several servers can share the same queues. In this configuration, sending a request to any of the servers adds the request to the shared queue. Whenever a server is ready to start processing a new request, it takes the next request from the shared queue, runs the action, and adds the results of the action back to the shared database so that they can be retrieved by any of the servers. You can therefore distribute requests between components without configuring a Distributed Action Handler (DAH).

NOTE:
You cannot use multiple servers to process a single request. Each request is processed by one server.

Prerequisites

- Supported databases:
  - PostgreSQL 9.0 or later.
  - MySQL 5.0 or later.
- If you use PostgreSQL, you must set the PostgreSQL ODBC driver setting `MaxVarChar` to **`0`** (zero). If you use a DSN, you can configure this parameter when you create the DSN. Otherwise, you can set the `MaxVarcharSize` parameter in the connection string.

Configure Connector Framework Server

To configure Connector Framework Server to use a shared action queue, follow these steps.

To store action queues in an external database

1. Stop Connector Framework Server, if it is running.
2. Open the Connector Framework Server configuration file.
3. Find the relevant section in the configuration file:
   - To store queues for all asynchronous actions in the external database, find the `[Actions]` section.
   - To store the queue for a single asynchronous action in the external database, find the section that configures that action.
4. Set the following configuration parameters.

`AsyncStoreLibraryDirectory`	The path of the directory that contains the library to use to connect to the database. Specify either an absolute path, or a path relative to the server executable file.
`AsyncStoreLibraryName`	The name of the library to use to connect to the database. You can omit the file extension. The following libraries are available: • **`postgresAsyncStoreLibrary`** - for connecting to a PostgreSQL database. • **`mysqlAsyncStoreLibrary`** - for connecting to a MySQL database.

`ConnectionString`	The connection string to use to connect to the database. The user that you specify must have permission to create tables in the database. For example:
	`ConnectionString=DSN=my_ASYNC_QUEUE`
	or
	`ConnectionString=Driver={PostgreSQL}; Server=10.0.0.1; Port=9876; Database=SharedActions; Uid=user; Pwd=password; MaxVarcharSize=0;`

   For example:

```
[Actions]
AsyncStoreLibraryDirectory=acidlls
AsyncStoreLibraryName=postgresAsyncStoreLibrary
ConnectionString=DSN=ActionStore
```

5. If you are using the same database to store action queues for more than one type of component, set the following parameter in the `[Actions]` section of the configuration file.

`DatastoreSharingGroupName`	The group of components to share actions with. You can set this parameter to any string, but the value must be the same for each server in the group. For example, to configure several Connector Framework Servers to share their action queues, set this parameter to the same value in every Connector Framework Server configuration. Micro Focus recommends

setting this parameter to the name of the component.

CAUTION:
Do not configure different components (for example, two different types of connector) to share the same action queues. This will result in unexpected behavior.

For example:

```
[Actions]
...
DatastoreSharingGroupName=ComponentType
```

6. Save and close the configuration file.

   When you start Connector Framework Server it connects to the shared database.

Store Action Queues in Memory

Connector Framework Server provides asynchronous actions. Each asynchronous action has a queue to store requests until threads become available to process them. These queues are usually stored in a datastore file or in a database hosted on a database server, but in some cases you can increase performance by storing these queues in memory.

NOTE:
Storing action queues in memory improves performance only when the server receives large numbers of actions that complete quickly. Before storing queues in memory, you should also consider the following:

- The queues (including queued actions and the results of finished actions) are lost if Connector Framework Server stops unexpectedly, for example due to a power failure or the component being forcibly stopped. This could result in some requests being lost, and if the queues are restored to a previous state some actions could run more than once.
- Storing action queues in memory prevents multiple instances of a component being able to share the same queues.
- Storing action queues in memory increases memory use, so please ensure that the server has sufficient memory to complete actions and store the action queues.

If you stop Connector Framework Server cleanly, Connector Framework Server writes the action queues from memory to disk so that it can resume processing when it is next started.

To configure Connector Framework Server to store asynchronous action queues in memory, follow these steps.

To store action queues in memory

1. Stop Connector Framework Server, if it is running.
2. Open the Connector Framework Server configuration file and find the `[Actions]` section.
3. If you have set any of the following parameters, remove them:

- `AsyncStoreLibraryDirectory`
- `AsyncStoreLibraryName`
- `ConnectionString`
- `UseStringentDatastore`

4. Set the following configuration parameters.

`UseInMemoryDatastore`	A Boolean value that specifies whether to keep the queues for asynchronous actions in memory. Set this parameter to **`TRUE`**.
`InMemoryDatastoreBackupIntervalMins`	(Optional) The time interval (in minutes) at which the action queues are written to disk. Writing the queues to disk can reduce the number of queued actions that would be lost if Connector Framework Server stops unexpectedly, but configuring a frequent backup will increase the load on the datastore and might reduce performance.

For example:

```
[Actions]
UseInMemoryDatastore=TRUE
InMemoryDatastoreBackupIntervalMins=30
```

5. Save and close the configuration file.

   When you start Connector Framework Server, it stores action queues in memory.

Use XSL Templates to Transform Action Responses

You can transform the action responses returned by Connector Framework Server using XSL templates. You must write your own XSL templates and save them with either an `.xsl` or `.tmpl` file extension.

After creating the templates, you must configure Connector Framework Server to use them, and then apply them to the relevant actions.

To enable XSL transformations

1. Ensure that the `autnxslt` library is located in the same directory as Connector Framework Server. If the library is not included in your installation, you can obtain it from Micro Focus Support.
2. Open the Connector Framework Server configuration file in a text editor.
3. In the `[Server]` section, ensure that the `XSLTemplates` parameter is set to **`true`**.

   CAUTION:
   If `XSLTemplates` is set to **`true`** and the `autnxslt` library is not present in the same directory as the configuration file, the server will not start.

4. (Optional) In the `[Paths]` section, set the `TemplateDirectory` parameter to the path to the directory that contains your XSL templates. The default directory is `acitemplates`.

5. Save and close the configuration file.
6. Restart Connector Framework Server for your changes to take effect.

To apply a template to action output

- Add the following parameters to the action:

`Template`	The name of the template to use to transform the action output. Exclude the folder path and file extension.
`ForceTemplateRefresh`	(Optional) If you modified the template after the server started, set this parameter to **`true`** to force the ACI server to reload the template from disk rather than from the cache.

  For example:

```
action=QueueInfo&QueueName=Ingest
               &QueueAction=GetStatus
               &Token=...
               &Template=myTemplate
```

  In this example, Connector Framework Server applies the XSL template `myTemplate` to the response from an `Ingest` action.

  NOTE:
  If the action returns an error response, Connector Framework Server does not apply the XSL template.

Example XSL Templates

Connector Framework Server includes the following sample XSL templates, in the `acitemplates` folder:

XSL Template	Description
`LuaDebug`	Transforms the output from the `LuaDebug` action, to assist with debugging Lua scripts.

Chapter 5: Ingest Data

This section describes how to send data to CFS.

Ingest Data using Connectors

To configure a connector to send data to CFS, follow these steps.

To configure a connector to send data to CFS

1. Stop the connector, if it is running. For information about how to stop a connector, refer to the connector's documentation.
2. Open the connector's configuration file in a text editor.
3. In the `[Ingestion]` section, set the following parameters:

`EnableIngestion`	To enable ingestion, set this parameter to **`true`**.
`IngesterType`	To send data to CFS, set this parameter to **`CFS`**.
`IngestHost`	The host name or IP address of the CFS.
`IngestPort`	The ACI port of the CFS.

   For example:

```
[Ingestion]
EnableIngestion=True
IngesterType=CFS
IngestHost=localhost
IngestPort=7000
```

4. Save and close the configuration file.

   You can now start the connector.

Ingest an IDX File

You can ingest an IDX file using the `Ingest` action.

Use the `adds` parameter to specify the document that you want to ingest. This parameter takes XML like the following example which ingests `c:\data.idx`:

```
<adds>
   <add>
      <source filename="c:\data.idx" />
   </add>
</adds>
```

The XML must be URL encoded:

```
http://server:port/action=ingest&adds=[URL encoded XML]
```

For more information about the `Ingest` action, refer to the *Connector Framework Server Reference*.

Ingest XML

Many systems export information in XML format and CFS has features to help you convert XML into IDOL documents.

NOTE:
The XML must be encoded in UTF-8.

You can configure CFS to transform XML files, with an XSL transformation, before they are processed. This is an optional step but can be useful in cases where your XML files do not resemble IDOL documents or you are processing XML from many sources and the files have different schemas. You can configure any number of transformations and CFS runs the first transformation where the ingested XML matches the specified schema. You can also configure a default transformation that CFS runs when an XML file does not match any of your schemas.

After an XML file has been transformed, or when transformation is not configured, CFS attempts to convert the XML into IDOL documents. The XML is parsed according to the rules that you configure in the `[XmlParsing]` section of the CFS configuration file. If the conversion is successful, the resulting metadata-only documents are added to the ingest queue (for more information about the ingestion process, see The Ingestion Process, on page 11). If the conversion does not result in any IDOL documents but the XML was transformed after matching a schema, CFS does not consider this as a failure and does not index any documents. Otherwise, for example if the XML is invalid, the XML file is added to the import queue so that it is processed by KeyView along with other file types.

Transform XML Files

CFS can transform XML files before attempting to parse them. XSL transformations are configured in the `[XmlTransformation]` section of the CFS configuration file.

To run a single transformation, you can specify the settings in the `[XmlTransformation]` section:

```
[XmlTransformation]
ValidationSchema=schema.xsd
TransformationStylesheet=transform.xslt
```

In this example, CFS uses the stylesheet `transform.xslt` to transform any XML file that matches `schema.xsd`.

If you are processing XML files that have more than one schema, you might want to configure several transformations. To do this, use the `Sections` parameter to specify the names of sections that configure the transformations:

```
[XmlTransformation]
Sections=XmlTransform1,XmlTransform2

[XmlTransform1]
ValidationSchema=schema1.xsd
TransformationStylesheet=transform1.xslt

[XmlTransform2]
TransformationStylesheet=transform2.xslt
```

In this example, any XML file that matches `schema1.xsd` is transformed by `transform1.xslt`. These files are then parsed. The parameter `ValidationSchema` is not set in the section `XmlTransform2`, so any files that do not match `schema1.xsd` are transformed by `transform2.xslt`.

You can configure as many different transformations as you require. If you set the parameter `ValidationSchema` in every section and an XML file does not match any of the schemas, it is not transformed.

Parse XML into Documents

CFS attempts to parse any XML file that it receives according to rules that are specified in the `[XMLParsing]` section of its configuration file. The parameters in the `[XMLParsing]` section specify:

- How to divide the XML into documents.
- How to populate each document's `DREREFERENCE` field.
- How to populate each document's `DRECONTENT` field.

To configure settings for parsing XML

1. Open the CFS configuration file.
2. In the `[XMLParsing]` section, set the following parameters:

`DocumentRootPaths`	A comma-separated list of paths to nodes that contain a single document. Specify the paths relative to the root of the XML. Use a forward slash (/) to represent levels in the XML hierarchy. Any elements contained within the specified node are added to the document as metadata.
`IncludeRootPath`	A Boolean value (default `false`) that specifies whether to include the node specified by `DocumentRootPaths` in the document. You might set this parameter to **`TRUE`** if the root node has attributes that you need to include in the document.
`ReferencePaths`	A comma-separated list of possible paths to a node that contains the

	document reference. Specify the paths relative to the node identified by `DocumentRootPaths`. Use a forward slash (/) to represent levels in the XML hierarchy. The XML for each document must contain exactly one node that matches the specified path(s).
`ContentPaths`	A comma-separated list of possible paths to a node that contains the document content. Specify the paths relative to the node identified by `DocumentRootPaths`. Use a forward slash (/) to represent levels in the XML hierarchy. If multiple content nodes are identified for a single document, a document is produced with multiple sections.

3. Save and close the configuration file.

Example

Consider the following XML:

```
<xml>
   <documents>
      <document>
         <metadata>
           <name>This is the name of the document</name>
           <created>28/02/15 11:01:17</created>
           <modified>28/02/15 15:23:00</modified>
         </metadata>
         <content>Here is some content</content>
       </document>
       <document>
         <metadata>
           <name>This is another document</name>
           <created>01/03/15 12:21:13</created>
           <modified>02/03/15 13:23:03</modified>
         </metadata>
         <different_content>Here is some content</different_content>
       </document>
     </documents>
 </xml>
```

To ingest this XML file, you might use the following configuration:

```
[XMLParsing]
DocumentRootPaths=documents/document
ReferencePaths=metadata/name
ContentPaths=content,different_content
```

To ingest the XML, send the ingest action to CFS:

```
http://localhost:7000/action=ingest&adds=%3Cadds%3E%3Cadd%3E%3Csource%20
                                         filename%3D%22xmlfile.xml%22%20
                                         lifetime%3D%22permanent%22%20%2F%3E
                                         %3C%2Fadd%3E%3C%2Fadds%3E
```

This would produce the following documents:

```
#DREREFERENCE This is the name of the document
#DREFIELD UUID="bfa1a8aac0b772d1ee467d830fa179bc"
#DREFIELD DocTrackingId="3cd0e5cf3160163adf7445d013ef10b1"
#DREFIELD ImportVersion="1207655"
#DREFIELD KeyviewVersion="10220"
#DREFIELD metadata/created="28/02/15 11:01:17"
#DREFIELD metadata/modified="28/02/15 15:23:00"
#DRECONTENT
Here is some content
#DREENDDOC

#DREREFERENCE This is another document
#DREFIELD UUID="aadf6628fccd0c6b885a79e2e39f4357"
#DREFIELD DocTrackingId="66a63287d85b500159c5b5fb099b99a5"
#DREFIELD ImportVersion="1207655"
#DREFIELD KeyviewVersion="10220"
#DREFIELD metadata/created="01/03/15 12:21:13"
#DREFIELD metadata/modified="02/03/15 13:23:03"
#DRECONTENT
Here is some content
#DREENDDOC
```

Ingest PST Files

Consider the following points before ingesting Microsoft Outlook Personal Folders (PST) files:

- The best results are usually obtained when KeyView uses MAPI to extract and filter PST files. To use MAPI, you must:
  - Run CFS on Windows.
  - Install Microsoft Outlook on the same machine as CFS. If you are using 64-bit CFS, install 64-bit Outlook. If you are using 32-bit CFS, install 32-bit Outlook.
  - Ensure that MAPI has write access to the PST files. Set the `WorkingDirectory` parameter in the `[ImportService]` section of the CFS configuration file so that CFS copies files to a working directory and processes the copies, rather than processing the files in their original location.
- PST files can contain a large amount of data and KeyView might not finish processing them within the default time limit allowed by CFS. Consider increasing the value of the `KeyviewTimeout` parameter, in the `[ImportService]` section of the CFS configuration file.

Ingest Password-Protected Files

To process password-protected files you must provide CFS with the passwords.

To specify the passwords for password-protected files

1. Create a credentials file to contain the passwords for your password-protected files:
    a. Open a text editor and create a new text file.
    b. Create an `[ImportService]` section in the file.
    c. In the `ImportService` section, set the following parameter:

`ImportCredentialCount`	The total number of file name and password combinations specified in the credentials file.

    For example:

    ```
    [ImportService]
    ImportCredentialCount=1
    ```

    d. Create a new section in the file, named `[Credential`*N*`]`, where *N* is the number of the file name/password combination, starting from 0.

    In the new section, set the following parameters:

`FileSpec`	The name of the password protected file(s). You can use the `*` wildcard to match the file name(s).
`Password`	The password for the file(s). You can encrypt the password using the password encryption utility. For information about how to do this, see Encrypt Passwords, on page 23.
`UserName`	The user name to use to open the file(s). Set this parameter if a user name is required to access the file.
`NotesIDFile`	The path of the ID file. Set this parameter for `.nsf` files only.

    For example, the following settings could be used to specify a single password for all ZIP files:

    ```
    [ImportService]
    ImportCredentialCount=1

    [Credential0]
    FileSpec=*.zip
    Password=9t3M3t7awt/J8A
    ```

    e. To specify further file name and password combinations, repeat steps c and d.
    f. Save the file to a suitable location.
2. Specify the location of the credentials file. There are several ways to do this:
    - To use the credentials file you created for all ingested documents, set the CFS configuration parameter `ImportCredentialFile` to the path of the file. For more information about this parameter, refer to the *Connector Framework Server Reference*.
    - To use the credentials file that you created to process a single document, set the document field `AUTN_CREDENTIALS`. This field accepts either the path to the credentials file, or the credentials file content. You can encrypt the text using the password encryption utility. The `AUTN_CREDENTIALS` field is removed from all documents before they are indexed. When you send an `ingest` action to CFS, you can set this field using the `xmlmetadata` element in the

`adds` or `updates` action parameter. For more information about the ingest action, refer to the *Connector Framework Server Reference*.

Ingest Data for Testing

To ingest data for testing purposes, use the `IngestTest` action. You can use this action to view the output of the ingestion process for a small amount of data, without the data being indexed into IDOL.

TIP:
CFS includes an XSL template to help you send `IngestTest` actions. Open a web browser and navigate to `http://`*`host`*`:7000/action=IngestTest&Template=IngestTest` (where *`host`* is the machine where CFS is running and `7000` is the CFS ACI port).

Micro Focus does not support the XSL template, it is provided only as an example of a template that you could build.

The `IngestTest` action has the following parameters:

```
/action=IngestTest
             &config=[base64_encoded_config]
             &adds=[URL_encoded_adds_xml]
```

`IngestTest` is similar to the `Ingest` action, but has the following differences which make it suitable for testing:

- `IngestTest` is a synchronous action, and the document data is returned in the ACI response.
- Indexing, whether as a result of ingestion or as a result of an import task, is disabled.
- Update and Delete commands are disabled (you cannot use the `updates` and `removes` action parameters like you can with the `Ingest` action).
- Any writer tasks that have been configured (`IdxWriter`, `XmlWriter`, `JsonWriter`, `CsvWriter`, `SqlWriter`) are disabled.
- Logging to the import log stream is disabled. The log messages are redirected to the action response.
- The global Lua variable `is_test` is set to `true`. You can use this variable in your Lua scripts to prevent certain parts of your scripts from running when you use the `IngestTest` action.

For more information about the `IngestTest` action and its parameters, refer to the *Connector Framework Server Reference*.

Chapter 6: Filter Documents and Extract Subfiles

CFS automatically extracts metadata, content, and sub-files from all files that are ingested. KeyView does not need to be configured, but this section describes how to customize the filtering and extraction process.

Customize KeyView Filtering

If necessary, you can customize the filtering and extraction process. For example, you can choose whether to extract comments added by reviewers to a Microsoft Word document.

To customize filtering, use the Import Service parameters, in the `[ImportService]` section of the CFS configuration file. For information about the parameters that you can set, refer to the *Connector Framework Server Reference*.

You can also customize KeyView filtering by modifying the configuration parameters in the KeyView `filters\formats.ini` configuration file. For more information about customizing KeyView filtering by modifying `formats.ini`, refer to the KeyView documentation.

Disable Filtering or Extraction for Specific Documents

To prevent KeyView from processing specific documents, you can add the following fields to documents. You can add the fields with any value.

AUTN_FILTER_META_ONLY	Prevents CFS extracting content from a file. CFS only extracts metadata and adds this information to the document.
AUTN_NO_FILTER	Prevents CFS extracting any text (metadata or content) from a file. This can be useful if you do not want to extract text from certain file types.
AUTN_NO_EXTRACT	Prevents CFS from extracting sub-files. This can be useful if you want to avoid extracting items from ZIP files and other container files.

NOTE:
To add a field to a document, use a Lua script. You must run the Lua script using a *Pre* import task. This is because *Post* import tasks run after KeyView filtering.

Related Topics

- Write and Run Lua Scripts, on page 52
- Add a Field to a Document, on page 59

Chapter 7: Manipulate and Enrich Documents

This section describes how to manipulate and enrich documents using CFS.

Introduction

The documents produced by connectors and CFS contain information extracted from the source repository. In many cases you might want to add additional information to documents, or modify the structure of the documents, before they are indexed.

To modify documents before they are indexed, use *Import Tasks* and *Index Tasks*. These are customizable processing tasks that you can run on documents. You can use these tasks to write documents to disk, manipulate documents, reject documents, and run custom Lua scripts.

Write documents to disk

You can write documents to disk in IDX or XML format. This allows you to view the information that is being indexed, so that you can check the information is being indexed as you expected. If necessary, you can then use other import tasks to manipulate and enrich the information.

Manipulate and enrich documents

You can use import tasks to enrich documents. For example, you can:

- extract the meaningful content from HTML, and discard advertisements, headers, and sidebars.
- divide document content into sections. Dividing a document can result in more relevant query results, because IDOL can return a specific part of a document in response to a query.
- extract speech from audio and video files, and write a transcription of the speech to the document content. IDOL Server can then use the speech for retrieval, clustering, and other operations.

Validate and reject documents

You can reject documents that you do not want to index, for example those that do not appear to contain valid content. When a document is rejected, it is not processed further and is not indexed. However, you can index the document into an IDOL Server that has been configured to handle failed documents.

Run a Lua Script

Lua is an embedded scripting language that you can use to manipulate documents and define custom processing rules. CFS includes Lua functions for manipulating documents and running other tasks.

Choose When to Run a Task

Import Tasks run when new documents are processed by CFS, before the documents are indexed. You can run Import Tasks before and/or after KeyView filtering.

- *Pre*-import tasks run before KeyView filtering. At this point the document only contains metadata extracted from the repository by the connector.
- *Post*-import tasks run after KeyView filtering. At this point the document also contains any content and metadata that was extracted from the file associated with the document.

Index Tasks run when a document's metadata (but not its content) is updated, or when a document is deleted. When a connector detects that document metadata has been updated or that a document has been deleted from a repository, it sends this information to CFS so that the document can be updated or removed from indexes such as IDOL Server.

- *Update* index tasks run when a document's metadata (but not its content) is updated.
- *Delete* index tasks run when a document is deleted from a repository.

You can run some tasks, such as the `Lua` task, at any point during the import or indexing process.

You can run other tasks only at specific points within the import or indexing process. For example, to validate the content of documents you must use a post-import task. You cannot use a pre-import task because pre-import tasks occur before KeyView filtering, when documents do not contain any content.

The following table shows when you can run each type of task.

Task	Import Tasks		Index Tasks	
	Pre	Post	Update	Delete
Run a Lua script				
Lua	✓	✓	✓	✓
Write documents to disk				
CsvWriter	✓	✓	✓	✓
IdxWriter	✓	✓	✓	✓
JsonWriter	✓	✓	✓	✓
SqlWriter	✓	✓	✓	✓
XmlWriter	✓	✓	✓	✓
Manipulate and enrich documents				
Eduction		✓		
EmailAddressNormalisation	✓	✓		
ExtractMetadata	✓			
HtmlExtraction	✓			
ImportFile	✓	✓		
Sectioner		✓		
Standardizer	✓	✓		
TextToDocs	✓			
Validate and reject documents				
BadFilesFilter		✓		
BinaryFileFilter		✓		

Task	Import Tasks		Index Tasks	
	Pre	Post	Update	Delete
ImportErrorFilter		✓		
SymbolicContentFilter		✓		
WordLengthFilter		✓		
Media analysis				
MediaServerAnalysis	✓	✓		
IdolSpeech	✓	✓		

You can also call many of the tasks from a Lua script, which allows more advanced processing. For example, you might want to run a task only on selected documents. For information about the Lua functions that are provided by CFS, refer to the *Connector Framework Server Reference*.

Related Topics

- The Import Process, on page 13.

Create Import and Index Tasks

Import tasks are configured in the `[ImportTasks]` section of the CFS configuration file. Import tasks run when files are imported, for example when a new item is retrieved from a repository or when the content of a file in a repository is updated. Use the `Pre` parameter to specify a list of tasks to run before KeyView filtering, and the `Post` parameter to specify a list of tasks to run after KeyView filtering.

Index tasks are configured in the `[IndexTasks]` section of the CFS configuration file. Use the `Update` parameter to specify a list of tasks to run when a connector instructs CFS to update the metadata of a document. Use the `Delete` parameter to specify a list of tasks to run when a connector instructs CFS to delete a document from indexes such as IDOL Server.

The tasks that you define run in sequence. In the following example, CFS creates an IDX file, then runs a Lua script, and then creates another IDX file:

```
[ImportTasks]
Post0=IdxWriter:C:\IDXArchive\before_script.idx
Post1=Lua:C:\Scripts\my_script.lua
Post2=IdxWriter:C:\IDXArchive\after_script.idx
```

To create an import task

1. Stop CFS.
2. Open the CFS configuration file.
3. Find the `[ImportTasks]` section of the configuration file, or create it if it does not exist.

4. Add the task by setting the `Pre` or `Post` parameter.

   The value of the `Pre` or `Post` parameter must be the name of the task that you want to run. Some tasks also require further information, such as the name of a file or the name of a section in the configuration file.

   For example, to run a Lua script before KeyView filtering:

   ```
   [ImportTasks]
   Pre0=Lua:myscript.lua
   ```

5. Some import tasks require you to identify the documents to process by adding a field to the documents. For example, the `IdolSpeech` task only runs on documents that have the `AUTN_NEEDS_TRANSCRIPTION` field. For more information about the document fields that are used with import tasks, see Document Fields for Import Tasks, below.

   To add a field to the documents that you want to process, use a Lua script. In the following example, a Lua script named `Filter.Lua` runs before an `IdolSpeech` import task, to identify suitable documents and add the field `AUTN_NEEDS_TRANSCRIPTION`.

   ```
   [ImportTasks]
   Pre0=Lua:Filter.lua
   Pre1=IdolSpeech:IdolSpeechSettings
   ```

6. Save the configuration file and restart CFS.

To create an index task

1. Stop CFS.
2. Open the CFS configuration file.
3. Find the `[IndexTasks]` section of the configuration file, or create it if it does not exist.
4. Add the task by setting the `Update` or `Delete` parameter.

   The value of the `Update` or `Delete` parameter must be the name of the task that you want to run. Some tasks also require further information, such as the name of a file or the name of a section in the configuration file.

   For example:

   ```
   [IndexTasks]
   Update0=Lua:myscript.lua
   ```

5. Save the configuration file and restart CFS.

Document Fields for Import Tasks

You can customize how documents are processed by import tasks, by adding the following fields to your documents.

> **NOTE:**
> The Lua script that adds the document fields must run before the import tasks.

AUTN_NEEDS_TRANSCRIPTION

To use an IDOL Speech Server to extract the speech from a document that represents an audio or video file, you must add the field `AUTN_NEEDS_TRANSCRIPTION` to the document. The `IdolSpeech` task only runs on documents that have this field. The field can have any value. For more information about the `IdolSpeech` task, see Analyze Speech, on page 68.

AUTN_FORMAT_CORRECT_FOR_TRANSCRIPTION

To bypass the transcoding step of an `IdolSpeech` task, add the field `AUTN_FORMAT_CORRECT_FOR_TRANSCRIPTION`. The field can have any value. Documents that have this field are not sent to a Transcode Server. For more information about the `IdolSpeech` task, see Analyze Speech, on page 68.

AUTN_AUDIO_LANGUAGE

To bypass the language identification step of an `IdolSpeech` task add the field `AUTN_AUDIO_LANGUAGE`. The value of the field must be the name of the IDOL Speech Server language pack to use for extracting speech. Documents that have this field are not sent to the IDOL Speech Server for language identification. For more information about the `IdolSpeech` task, see Analyze Speech, on page 68.

AUTN_NEEDS_MEDIA_SERVER_ANALYSIS

To perform analysis on media files using the `MediaServerAnalysis` task, you must add this field to every document that you want to analyze. The field can have any value.

Write and Run Lua Scripts

Connector Framework Server supports Lua, an embedded scripting language. CFS supports all standard Lua functions. For more information about Lua, refer to http://www.lua.org/.

You can use a Lua script to:

- Add or modify document fields.
- Run built-in processing tasks, such as Eduction or image analysis.
- Call out to an external service, for example to alert a user.
- Interface with other libraries.

Write a Lua Script

Your Lua script must have the following structure:

```
function handler(document)
   ...
end
```

The `handler` function is called for each document and is passed a document object. The document object is an internal representation of the document being processed. Modifying this object changes the document.

For CFS to continue processing the document, the function must return **`true`**. If the function returns **`false`**, the document is discarded.

The script can also terminate due to an error, for example if you use the Lua `error` function or call a Lua function that causes an error. In this case CFS continues to process the document, but places an error message in the `ImportErrorDescription` field.

TIP:
You can write a library of useful functions to share between multiple scripts, which you can then include in the scripts by adding **`dofile("library.lua")`** to the top of the lua script outside of the `handler` function.

Run a Lua Script

To run a Lua script, create a `Lua` import or index task, and specify the path to your script. You can run Lua scripts using *pre* and *post* Import Tasks, and using *update* and *delete* index tasks. For example:

```
[ImportTasks]
Post0=Lua:c:\scripts\script1.lua
```

Debug a Lua Script

When you run a Lua script and the script fails due to an error, CFS writes the error to the import log stream, and to the `ImportErrorDescription` field of any documents that are affected.

To debug your Lua scripts, you can use the `LuaDebug` action. You can use this action to pause and resume scripts, and set and remove breakpoints. When a script is paused you can view the values of variables, view a stack trace, and step over single lines.

Sessions

CFS can have more than one import thread, and might run multiple Lua scripts concurrently. This means that you can have multiple Lua Debugging sessions. You might want to pause or continue running scripts on one thread but not others. Some of the commands available through the `LuaDebug` action allow or require you to specify a `session` action parameter. If the `session` parameter is optional and you do not specify a session, the command applies to all sessions. To view open sessions and obtain the values you can set for the `session` action parameter, use the command `/action=LuaDebug&command=get-status`.

Example

The following procedure demonstrates how to set a breakpoint in a script, view the values of Lua variables when the script is paused, and step over single lines. The actions in this procedure assume that your CFS is running on the local machine and is listening for actions on port 7000. For more information about the `LuaDebug` action, refer to the *Connector Framework Server Reference*.

To debug a Lua script

1. In the CFS configuration file, configure the script to run. This example uses the `AddLanguageDetectionFields` script that is included with CFS. For example:

   ```
   [ImportTasks]
   Post0=Lua:scripts/AddLanguageDetectionFields.lua
   ```

2. Start CFS.
3. To pause the script before a specific line is executed, set a breakpoint on that line. Line 33 of the `AddLanguageDetectionFields` script sends an action to IDOL Server and stores the response in a variable named `response`. To stop the script before this happens, use the following action:

   ```
   http://localhost:7000/action=luadebug
                               &command=set-breakpoint
                               &file=scripts/AddLanguageDetectionFields.lua
                               &line=33
   ```

4. (Optional) Confirm the breakpoint has been set using the `get-breakpoints` command:

   ```
   http://localhost:7000/action=luadebug&command=get-breakpoints
   ```

   CFS returns the response.

   ```
   <autnresponse xmlns:autn="http://schemas.autonomy.com/aci/">
      <action>LUADEBUG</action>
      <response>SUCCESS</response>
      <responsedata>
         <data>
            <command>get-breakpoints</command>
            <breakpoints>
               <breakpoint
   source="C:\Autonomy\ConnectorFramework\scripts\AddLanguageDetectionFields.lua"
   line="33"/>
            </breakpoints>
         </data>
      </responsedata>
   </autnresponse>
   ```

5. Send CFS an `IngestTest` action so that CFS ingests a document and runs the script:

   ```
   http://localhost:7000/action=IngestTest&adds=...
   ```

   TIP:
   Use an `IngestTest` action, rather than an `Ingest` action, because the `IngestTest` action does not index any information into IDOL Server. For more information about the `IngestTest` action, see Ingest Data for Testing, on page 44.

   CFS runs the script. The `IngestTest` action does not finish (because the script is paused at the breakpoint) and therefore does not return a response.

6. Retrieve a token for the debugging session by sending CFS the `LuaDebug` command `get-status`:

```
http://localhost:7000/action=LuaDebug&command=get-status
```

CFS returns the response. You can see that there is a single debugging session and the Lua script has stopped at the breakpoint.

```
<autnresponse xmlns:autn="http://schemas.autonomy.com/aci/">
   <action>LUADEBUG</action>
   <response>SUCCESS</response>
   <responsedata>
      <data>
         <command>get-status</command>
         <session id="e4f7c45f561930cd17a5aed0fe1481d8">
            <status>AtBreak</status>
         </session>
      </data>
   </responsedata>
</autnresponse>
```

7. To retrieve the values of the Lua variables at the breakpoint, run the `LuaDebug` command `get-locals`. Use the session token that you retrieved with the `get-status` command:

```
http://localhost:7000/action=LuaDebug
                           &command=get-locals
                           &session=e4f7c45f561930cd17a5aed0fe1481d8
```

CFS returns the response.

```
<autnresponse xmlns:autn="http://schemas.autonomy.com/aci/">
  <action>LUADEBUG</action>
  <response>SUCCESS</response>
  <responsedata>
    <data>
      <command>get-locals</command>
      <session id="e4f7c45f561930cd17a5aed0fe1481d8">
        <locals>
          ...
          <local name="idolHost" type="string">localhost</local>
          <local name="idolACIPort" type="number">9000</local>
          <local name="timeout" type="number">30000</local>
          ...
          ...
          <local type="string" name="detectionString">This is a document that
               contains text in English. Automatic Language Detection will
               detect the language and add the information to the document
          ...</local>
        </locals>
      </session>
    </data>
  </responsedata>
</autnresponse>
```

8. Use the `step` command to run line 33. CFS does not run subsequent lines (the script will remain

paused). Use the session token you retrieved with the `get-status` command:

```
http://localhost:7000/action=LuaDebug
                           &command=step
                           &session=e4f7c45f561930cd17a5aed0fe1481d8
```

CFS returns the response.

```
<autnresponse xmlns:autn="http://schemas.autonomy.com/aci/">
  <action>LUADEBUG</action>
  <response>SUCCESS</response>
  <responsedata>
    <data>
      <command>step</command>
      <session id="e4f7c45f561930cd17a5aed0fe1481d8"/>
    </data>
  </responsedata>
</autnresponse>
```

9. To see what effect the step had on the variables, run the `get-locals` command again. You should see a new variable named `response` that contains the response from the `DetectLanguage` action.

```
http://localhost:7000/action=LuaDebug
                           &command=get-locals
                           &session=e4f7c45f561930cd17a5aed0fe1481d8
```

10. After examining the variables, you might want to remove the breakpoint. To remove the breakpoint, send CFS the following action:

```
http://localhost:7000/action=LuaDebug
                           &command=remove-breakpoint
                           &file=scripts/AddLanguageDetectionFields.lua
                           &line=33
```

CFS returns the response. You can also use `/action=LuaDebug&command=get-breakpoints` to confirm that the breakpoint has been removed.

11. To continue running the Lua script, use the `continue` command:

```
http://localhost:7000/action=LuaDebug
                           &command=continue
                           &session=e4f7c45f561930cd17a5aed0fe1481d8
```

CFS continues to run the script. The `IngestTest` action finishes and returns a response.

Lua Scripts Included With CFS

The CFS installation directory includes a `scripts` folder that includes the following Lua scripts:

Script	Description
`AddLanguageDetectionFields.lua`	Detects the language of a document's content, using the IDOL Server action `DetectLanguage`. The script then adds fields describing the language and encoding to the

Script	Description
	document's metadata. The script demonstrates how to: • send an action to an ACI server. • parse the action response to a `LuaXmlDocument`. • use the methods of `LuaXmlDocument` to extract data from the document. The script assumes that an IDOL Server is installed on the local machine with an ACI port of 9000. You might need to modify these values. If you use this script, run it as a *post* import task so that it runs after KeyView has extracted document content.
`CategorySuggestFromText.lua`	Sends a document to IDOL for categorization, and adds information about the matching categories to the document's metadata. For information about how to use this script, see Categorize Documents, on page 73.
`filterdodgyfiles.lua`	An example script that demonstrates various ways to reject unwanted files, for example by file extension, by detecting the file format, or by analyzing the file content.
`identifiers.lua`	Adds sub-file indexes to the `AUTN_IDENTIFIER` document field of sub-files. This allows a connector to retrieve the sub-file, rather than the whole container, when the `collect` or `view` actions are used to retrieve the original file. If you use this script, you must run it as a *post* import task (so that it runs after KeyView processes the documents). For more information about the `AUTN_IDENTIFIER` field, see AUTN_IDENTIFIER, on page 179.
`IdolSpeech.lua`	Runs speech-to text on all files identified by KeyView as containing audio or video. To use this script, you must configure the settings for your IDOL Speech Server in the `[IdolSpeechSettings]` section of the CFS configuration file. For more information about using this script, see Run Analysis on All Audio and Video Files, on page 69.
`mediaserver/*.lua`	These scripts run analysis on images, audio files, and video files by sending them to Media Server. For information about configuring media analysis, see Analyze Media, on page 62.

NOTE:
CFS also includes scripts for use with Eduction. Some of these scripts are Eduction post processing scripts, which modify the output from an Eduction import task. The post processing scripts have the entry point `function processmatch (edkmatch)`, rather than `function`

handler (document). You must run a post processing script using the Eduction import task. Do not run an Eduction post processing script using a Lua task. For more information about Eduction Lua Post Processing, see Lua Post Processing, on page 77. For information about the Eduction scripts that are included with CFS, refer to the *Eduction User Guide.*

Use Named Parameters

Some Lua functions have an argument that takes named parameters. This argument is a table in which you can specify values for various parameters that affect the operation of the function.

You can specify a value for every parameter, or just those that you need. If you do not specify a value for a parameter, the function uses a default value. You can also specify the name of a configuration section and the function will read settings from that section in the CFS configuration file.

For example, when you call the function `looks_like_language`, you can set only the `term_file` named parameter, and use default values for the other settings:

```
looks_like_language(document, { term_file = "english.ocr" })
```

You might choose to set the `stop_list` parameter as well:

```
looks_like_language(document, { term_file = "english.ocr",
                       stop_list = "englishstoplist.dat" })
```

Alternatively, you can specify the name of a section in the CFS configuration file:

```
looks_like_language(document, { term_file = "english.ocr",
                       section = "LanguageSettings" })
```

In this example, the function uses the `english.ocr` term file. The settings for the remaining parameters are read from the `LanguageSettings` section of the CFS configuration file.

If you specify the name of a configuration section and use named parameters, the named parameters override any values set in the configuration file. In the following example, the `threshold` is set to 100, while other parameters (like `term_file`) are read from the `LanguageSettings` section:

```
looks_like_language(document, { section = "LanguageSettings",
                       threshold=100 })
```

For information about individual named parameters and corresponding configuration parameters, refer to the *Connector Framework Server Reference*.

Enable or Disable Lua Scripts During Testing

Lua scripts run by CFS can read a global Lua variable, `is_test`.

- When a script runs as part of an `Ingest` action, this variable is `false`.
- When a script runs as part of an `IngestTest` action, this variable is `true`.

You can use the `is_test` variable to enable or disable parts of a script. For example:

```
if is_test then
   -- The part of the script to enable for IngestTest
   -- (or disable for Ingest)
```

```
end

if not is_test then
   -- The part of the script to disable for IngestTest
   -- (or enable for Ingest)
end
```

Example Lua Scripts

This section contains example Lua scripts.

Add a Field to a Document

The following script demonstrates how to add a field named “MyField” to a document, with a value of “MyValue”.

```
function handler(document)
   document:addField("MyField", "MyValue");
   return true;
end
```

The following script demonstrates how to add the field `AUTN_NEEDS_IMAGE_SERVER_ANALYSIS` to all JPEG, TIFF and BMP documents. This field specifies that the documents can be processed using an `ImageServerAnalysis` import task (you must also define the task in the CFS configuration file).

The script finds the file type using the `DREREFERENCE` document field, so this field must contain the file extension for the script to work correctly.

```
function handler(document)
   local extensions_for_ocr = { jpg = 1 , tif = 1, bmp = 1 };
   local filename = document:getFieldValue("DREREFERENCE");
   local extension, extension_found =
      filename:gsub("^.*%.(%w+)$", "%1", 1);

   if extension_found > 0 then
      if extensions_for_ocr[extension:lower()] ~= nil then
         document:addField("AUTN_NEEDS_IMAGE_SERVER_ANALYSIS", "");
      end
   end

   return true;
end
```

Count Sections

For each document, this Lua script adds a total sections count to the title, and replaces the content of each section with the section number.

```
function handler(document)
   local section_count = 0;
   local section = document;

   while section do
      section_count = section_count + 1;
      section:setContent("Section "..section_count);
      section = section:getNextSection();
   end

   local title = document:getFieldValue("TITLE");

   if title == nil then title = "" end
   document:setFieldValue("TITLE", title .." Total Sections "
      ..section_count);

   return true;
end
```

Merge Document Fields

This script demonstrates how to merge the values of document fields.

When you extract data from a repository, CFS can produce documents that have multiple values for a single field, for example:

```
#DREFIELD ATTACHMENT="attachment.txt"
#DREFIELD ATTACHMENT="image.jpg"
#DREFIELD ATTACHMENT="document.pdf"
```

This script shows how to merge the values of these fields, so that the values are contained in a single field, for example:

```
#DREFIELD ATTACHMENTS="attachment.txt, image.jpg, document.pdf"
```

Example Script

```
function handler(document)
   onefield(document,"ATTACHMENT","ATTACHMENTS")
   return true;
end

function onefield(document,existingfield,newfield)
   if document:hasField(existingfield) then
      local values = { document:getFieldValues(existingfield) }
      local newfieldvalue=""

      for i,v in ipairs(values) do
         if i>1 then
            newfieldvalue = newfieldvalue ..", "
         end

         newfieldvalue = newfieldvalue..v
      end

      document:addField(newfield,newfieldvalue)
   end

   return true;
end
```

Add Titles to Documents

IDOL documents have a field named `DRETITLE` that can contain a title for the document. Front end applications might use the value of this field to present a title to users when displaying query results.

You should not rely on a connector to add a document title, because the connector might not be able to obtain this information. A suitable title for an e-mail message could be the subject of the e-mail, but this is not extracted until the e-mail is processed by CFS.

You can therefore use a Lua script to add a title to documents that do not have one, and, if necessary, ensure that all documents have suitable titles.

CFS includes a Lua script that adds titles to documents. The script is named `ExtractDreTitles.lua`, and is located in the `scripts` folder, in the CFS installation directory. You can use this script or modify it to suit your requirements.

The unmodified script ensures that all documents have a title. If a title has already been added to the document, that title is respected. If the document does not have a title, the script attempts to extract one from metadata fields that are added by KeyView and often contain titles. If none of these fields are present, the script adds a title by extracting the original file name from the field `DREORIGINALNAME`.

To add titles to documents using the `ExtractDreTitles` Lua script

1. Open the CFS configuration file.
2. Find the `[ImportTasks]` section of the configuration file, or create this section if it does not exist.
3. In the `[ImportTasks]` section, configure a `Post` import task to run the Lua script `scripts/ExtractDreTitles.Lua`.

   For example:

   ```
   [ImportTasks]
   Post0=Lua:scripts/ExtractDreTitles.lua
   ```

   > **TIP:**
   > You must use a `Post` task so that the script runs after KeyView filtering.

4. Save and close the configuration file.

Analyze Media

Images, audio, and video are examples of unstructured information that represent a vast quantity of data. CFS extracts metadata from these files but cannot process their content, so by default documents that represent these files are indexed without any content.

To enrich documents that represent rich media files, you can send the files to an IDOL Media Server for analysis. Media Server can:

- extract text from scanned documents, and subtitles and scrolling text from video.
- identify people that appear by matching faces to a database of known faces.
- identify known logos and objects.
- detect and read barcodes, including QR codes.
- determine the language of speech in a video file, convert the speech into text, and identify any known speakers (speech processing also requires an IDOL Speech Server).

For more information about the types of analysis that you can run, refer to the *Media Server Administration Guide*.

> **NOTE:**
> Some types of analysis require you to train Media Server before you start processing.

Create a Media Server Configuration

To run analysis on media, you must create a Media Server configuration file that instructs Media Server how to process the media. Micro Focus recommends that you save the configuration in a location accessible to CFS, and configure CFS to send the configuration to Media Server with each request.

Example Media Server configurations are provided with CFS in the `script_resources/mediaserver` directory.

The Media Server configuration must meet the following requirements.

Ingestion

There is no single configuration that can process both images and video, so you must configure Media Server to ingest the correct type of media.

The following example demonstrates how to configure ingestion. To process audio or video files, set the parameter `IngestEngine=AudioVideo` so that Media Server uses the settings in the `[AudioVideo]` section of the configuration. To process image files (including PDF files and office documents that contain embedded images), set the parameter `IngestEngine=Image`. The Lua functions `analyze_media_in_document` and `analyze_media_in_file` allow you to override individual parameters, so if you request analysis from a Lua script you can first determine the file type and then set the value of the parameter when you call the function.

```
[Ingest]
IngestRate=0
```

```
IngestEngine=AudioVideo

[AudioVideo]
Type=LibAv

[Image]
Type=Image
```

For more information about configuring ingestion, and the file types that are supported, refer to the *Media Server Administration Guide*.

Analysis

Create a section in the configuration file named `[Analysis]`, and configure the analysis operations that you want to run.

The following example configures face detection and optical character recognition:

```
[Analysis]
AnalysisEngine0=FaceDetect
AnalysisEngine1=OCR

[FaceDetect]
Type=FaceDetect
MinSize=70

[OCR]
Type=OCR
```

For more information about configuring analysis in Media Server, refer to the *Media Server Administration Guide*.

Output

CFS expects Media Server to return the results of analysis in the `process` action response. You must create a section in the configuration file named `[Output]`, and configure an output task to write data to the action response.

```
[Output]
OutputEngine0=response
```

```
[response]
Type=response
```

Configure the Media Analysis Task

You can run media analysis on documents by using the `MediaServerAnalysis` import task. This task only processes documents that have the document field `AUTN_NEEDS_MEDIA_SERVER_ANALYSIS`, so you must add this field to any document that you want to process.

To configure the Media Server Analysis task

1. Write a Lua script to add the document field `AUTN_NEEDS_MEDIA_SERVER_ANALYSIS` to the documents that you want to analyze. For an example script that adds a field to a document, see Add a Field to a Document, on page 59.
2. Open the CFS configuration file.
3. In the `[ImportTasks]` section, configure a `Pre` or `Post` import task to run your Lua script. For example:

   ```
   [ImportTasks]
   Pre0=Lua:scripts/TagVideoFiles.lua
   ```

4. Add another `Pre` or `Post` task to run the `MediaServerAnalysis` task. Set the `Pre` or `Post` parameter to `MediaServerAnalysis`, followed by a colon (`:`), followed by the name of the section in the CFS configuration file that contains the task settings. For example:

   ```
   Pre1=MediaServerAnalysis:MediaServerSettings
   ```

5. Create a new section in the configuration file, using the name you specified in Step 4.
6. In the new section, set the following parameters:

`MediaServerHost`	The host name and ACI port of your Media Server. To distribute requests between several servers, specify a comma-separated list of servers.
`MediaAnalysisTransform`	(Optional) To transform the metadata produced by Media Server, before CFS adds the data to your documents, set this parameter to the path of the XSL transformation to use. By default, CFS adds the information to your documents in a document field named `MediaServerAnalysis`, in the same structure that is returned from Media Server.

7. Specify the Media Server configuration file that you want to use for running analysis:
   - If you saved your configuration file in the directory specified by the `ConfigDirectory` parameter, in the `[Paths]` section of the Media Server configuration file, set `MediaServerConfigurationName` to the name of the configuration.
   - If you saved your configuration file in a location accessible by CFS, set the parameter `MediaServerConfigurationFileName` to the path of the configuration file. If you set a relative path, specify the path relative to CFS, not relative to Media Server.
8. Specify how to send media to Media Server:
   - If your Media Server can read files directly from the CFS working directory, set `ReadFromOriginalLocation=TRUE`.
   - To copy files to a shared folder, set the configuration parameter `MediaServerSharedPath`. This folder must be accessible to both CFS and Media Server. CFS copies files to the shared folder so that Media Server can read them. Micro Focus recommends that you use a shared folder for sending large files.
   - To send files to Media Server using HTTP POST requests, set neither `ReadFromOriginalLocation` nor `MediaServerSharedPath`.
9. Save and close the configuration file.

Examples

The following example shows how to configure the `MediaServerAnalysis` task. This example runs analysis using a configuration named `RecognizeFacesInVideo` that exists on the Media Server machine:

```
[ImportTasks]
Pre0=Lua:TagVideoFiles.lua
Pre1=MediaServerAnalysis:MediaServerSettings

[MediaServerSettings]
MediaServerHost=localhost:14000
MediaServerConfigurationName=RecognizeFacesInVideo
ReadFromOriginalLocation=TRUE
```

The following example is similar but configures CFS to send a configuration file to Media Server:

```
[ImportTasks]
Pre0=Lua:TagVideoFiles.lua
Pre1=MediaServerAnalysis:MediaServerSettings

[MediaServerSettings]
MediaServerHost=localhost:14000
MediaServerConfigurationFileName=./script_resources/mediaserver/facerecognition.cfg
ReadFromOriginalLocation=TRUE
```

If your CFS and Media Server are running on separate machines, you can configure CFS to copy media files to a shared folder:

```
[ImportTasks]
Pre0=Lua:TagVideoFiles.lua
Pre1=MediaServerAnalysis:MediaServerSettings

[MediaServerSettings]
MediaServerHost=media1:14000,media2:14000
MediaServerConfigurationName=RecognizeFacesInVideo
MediaServerSharedPath=\\server\videofiles
```

CFS adds the results of analysis to your documents. By default, the information is added in the same structure that is returned from Media Server, in a document field named `MediaServerAnalysis`. Using the configuration parameter `MediaAnalysisTransform`, you can configure CFS to run an XSL transformation to transform the information before adding it a document:

```
[ImportTasks]
Pre0=Lua:TagVideoFiles.lua
Pre1=MediaServerAnalysis:MediaServerSettings

[MediaServerSettings]
MediaServerHost=media1:14000,media2:14000
MediaServerConfigurationName=RecognizeFacesInVideo
```

```
MediaServerSharedPath=\\server\videofiles
MediaAnalysisTransform=./xslt/transform.xsl
```

For more information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Run Analysis From Lua

CFS provides Lua functions to run media analysis from a Lua script. These functions are named `analyze_media_in_document` and `analyze_media_in_file`. There are several advantages to running media analysis from a Lua script, instead of using the `MediaServerAnalysis` import task.

Firstly, you can use a single configuration to process audio, video, and image files. Your Lua script can identify the type of content that is associated with a document, and choose the correct Media Server engine to use for ingesting that content. The only way to process audio, video, and image files using the `MediaServerAnalysis` import task is to configure several tasks.

Secondly, you can configure more complex operations. For example, you can write a Lua script that sends audio to Media Server for language identification, and then uses the results of language identification to run speech-to-text with the correct language pack.

Finally, you can run analysis from Lua by configuring a single import task. To use the `MediaServerAnalysis` import task, you run a Lua script that identifies the documents to process, followed by the `MediaServerAnalysis` task itself.

CFS is supplied with example scripts that run media analysis. The scripts are in the `scripts/mediaserver` folder, in the CFS installation directory.

The following procedure demonstrates how to configure media analysis from a Lua script, in this case language detection followed by speech-to-text.

To run media analysis from Lua

1. Write a Lua script that identifies the documents that you want to process and calls the function `analyze_media_in_document` (or `analyze_media_in_file`).

   An example Lua script for running language detection and speech-to-text is located at `./scripts/mediaserver/LangDetectAndSpeechToText.lua`.

2. Create one or more configurations for Media Server that specify the tasks to perform. The Lua script `LangDetectAndSpeechToText.lua` uses two configurations, one for language detection and another for speech-to-text:

   - `./script_resources/mediaserver/langdetect.cfg`
   - `./script_resources/mediaserver/speechtotext.cfg`

   If you are using the example configuration files, check that the details are correct for your environment. For example, you might need to set the host name and ACI port of your Speech Server.

3. In the CFS configuration file, create an import task to run the Lua script. For example:

   ```
   [ImportTasks]
   Pre0=Lua:./scripts/mediaserver/LangDetectAndSpeechToText.lua
   ```

```
[MediaServerSettings]
MediaServerHost=mediaserver:14000
ReadFromOriginalLocation=true
// MediaServerSharedPath=<Share Directory UNC path>
```

The example script passes the `[MediaServerSettings]` section to the Lua function `analyze_media_in_document`. In the example configuration, above, this section provides the host name and ACI port of the Media Server and specifies how Media Server can access the media.

You can provide files to Media Server in several ways:

- If your Media Server can read files directly from the CFS working directory, set `ReadFromOriginalLocation=TRUE`.
- To copy files to a shared folder, set the configuration parameter `MediaServerSharedPath`. This folder must be accessible to both CFS and Media Server. CFS copies files to the shared folder so that Media Server can read them. Micro Focus recommends that you use a shared folder for sending large files.
- To send files to Media Server using HTTP POST requests, set neither `ReadFromOriginalLocation` nor `MediaServerSharedPath`.

4. Save and close the configuration file.

Examples

The following example configuration runs OCR on all supported image and video files ingested by CFS:

```
[ImportTasks]
Pre0=Lua:scripts/mediaserver/OCR.lua

[MediaServerSettings]
MediaServerHost=localhost:14000
ReadFromOriginalLocation=TRUE
```

If your CFS and Media Server are running on separate machines, you can configure CFS to copy the files to a shared folder:

```
[ImportTasks]
Pre0=Lua:scripts/mediaserver/OCR.lua

[MediaServerSettings]
MediaServerHost=mediaserver:14000
MediaServerSharedPath=\\server\videofiles
```

Troubleshoot Media Analysis

This section describes how to troubleshoot problems that might occur when you configure media analysis.

Error: Failed to find output node in response

```
Task (type: POST) failed with error: MediaServerAnalysis task failed: Failed to
```

```
find output node in response
```

Analysis will fail if CFS cannot retrieve the results of analysis from Media Server. If your analysis task fails with this error, check that your Media Server configuration includes an output task to add the results of analysis to the `process` action response. For example:

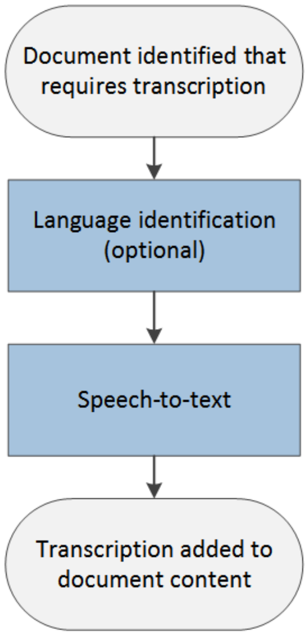
```
[Output]
OutputEngine0=response

[response]
Type=response
```

Analyze Speech

CFS extracts metadata from audio and video files but cannot process their content, so by default documents that represent audio and video are indexed without any content. You can enrich these documents by sending the files to an IDOL Speech Server. The Speech Server processes the audio, extracts any speech, and writes it to the document content.

The processing task that sends files to IDOL Speech Server for analysis is called `IdolSpeech`. It can include the following steps.


Document identified that requires transcription
Language identification (optional)
Speech-to-text
Transcription added to document content

1. Documents are identified that require speech-to-text processing.
2. (Optional) CFS sends the audio to an IDOL Speech Server to determine the language of the speech.
3. CFS sends the audio to an IDOL Speech Server for transcription.
4. CFS adds the transcription to the document content.

Run Analysis on All Audio and Video Files

To run speech-to-text on all files identified by KeyView as containing audio or video, run the Lua script `scripts/IdolSpeech.Lua`. The script reads settings from the `[IdolSpeechSettings]` section of the CFS configuration file.

The following example demonstrates how to run the script and specify information about your Speech Server:

```
[ImportTasks]
Pre0=Lua:scripts/IdolSpeech.lua

[IdolSpeechSettings]
IdolSpeechServers=server:15000
IdolSpeechLanguage=ENUK
```

The `IdolSpeechServers` parameter specifies the host name or IP address, and ACI port, of your Speech Server. Speech-to-text processing can be time consuming, so you can distribute the load over more than one Speech Server. For information about how to do this, see Use Multiple Speech Servers, on the next page.

The `IdolSpeechLanguage` parameter is optional and specifies the language pack to use for transcription. If you do not set this parameter, Speech Server runs language detection on each file and chooses a language pack automatically. If you know that all of your files are in the same language, Micro Focus recommends setting this parameter to reduce the load on the Speech Server.

If you prefer to send files to your IDOL Speech Server by writing them to a shared folder, add the `IdolSpechUseSharedPath` and `SharedPath` parameters to the configuration:

```
[ImportTasks]
Pre0=Lua:scripts/IdolSpeech.lua

[IdolSpeechSettings]
IdolSpeechServers=server:15000
IdolSpeechLanguage=ENUK
IdolSpeechUseSharedPath=true
SharedPath=\\server\SharedPath
```

Setting the parameter `IdolSpeechUseSharedPath` to `true` specifies that CFS sends files to Speech Server by copying them to a shared folder. The `SharedPath` parameter specifies the location of the shared folder. The folder must be accessible to both CFS and Speech Server.

Run Analysis on Specific Documents

To run speech-to-text on specific documents, you can modify the criteria in `scripts/IdolSpeech.lua`, or you can use the `IdolSpeech` import task and write your own Lua script to identify the documents to process. The `IdolSpeech` task only processes documents that have the field `AUTN_NEEDS_TRANSCRIPTION`, so your script must add this field to any document that you want to process.

The following example shows how to configure the `IdolSpeech` task in the CFS configuration file:

```
[ImportTasks]
Pre0=Lua:Identify_Audio_Files.lua
Pre1=IdolSpeech:IdolSpeechSettings

[IdolSpeechSettings]
IdolSpeechServers=server:15000
IdolSpeechLanguage=ENUK
```

The `Pre0` import task runs a Lua script that determines whether a file is suitable for transcription. You must write this script. The script must add the field `AUTN_NEEDS_TRANSCRIPTION` to any documents that you want to process. You can include conditions in the script to filter documents based on the document source, file type, or metadata extracted by KeyView.

The `Pre1` import task is the `IdolSpeech` task. It specifies the name of a section in the configuration file that contains the settings for the task. In this example the section is named `IdolSpeechSettings`.

The `IdolSpeechServers` parameter specifies the host name or IP address, and ACI port, of your IDOL Speech Server. To use multiple Speech Servers, see Use Multiple Speech Servers, below.

The `IdolSpeechLanguage` parameter is optional and specifies the language pack to use for transcription. You can set this parameter when all of your audio files are in the same language. If your audio files are in different languages, remove `IdolSpeechLanguage` so that Speech Server uses language detection to detect the language for each document. For more information about language identification, see Language Identification, on the next page.

If you prefer to send files to your IDOL Speech Server by writing them to a shared folder, add the `IdolSpeechUseSharedPath` and `SharedPath` parameters to the configuration:

```
[ImportTasks]
Pre0=Lua:Identify_Audio_Files.lua
Pre1=IdolSpeech:IdolSpeechSettings

[IdolSpeechSettings]
IdolSpeechServers=server:15000
IdolSpeechLanguage=ENUK
IdolSpeechUseSharedPath=true
SharedPath=\\server\SharedPath
```

Setting `IdolSpeechUseSharedPath=TRUE` instructs CFS to send files to Speech Server by writing them to the shared folder. specified by the `SharedPath` parameter.

For more information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Use Multiple Speech Servers

Language identification and speech-to-text processing can be time consuming. To increase performance, you can use several IDOL Speech Servers. The `IdolSpeechServers` configuration parameter accepts a comma-separated list of servers. For example:

```
IdolSpeechServers=server1:15000,server2:15000
```

Alternatively, you can use a numbered list:

```
IdolSpeechServers0=server1:15000
IdolSpeechServers1=server2:15000
```

Language Identification

To convert speech to text successfully, the IDOL Speech Server must know the language of the speech.

The IDOL Speech Server can automatically identify the language of speech. If language identification is not bypassed using one of the following methods it is performed automatically.

To bypass language identification

- To bypass language identification for all documents, set the `IdolSpeechLanguage` configuration parameter. This parameter specifies the language pack to use for all documents and takes precedence over other language settings. You can set this parameter when all of your audio is in the same language.
- To bypass language identification for a single document, add the field `AUTN_AUDIO_LANGUAGE` to the document. The value of the field must identify the language pack to use for transcription, for example:

  ```
  #DREFIELD AUTN_AUDIO_LANGUAGE="ENUK"
  ```

For a list of IDOL Speech Server language packs, refer to the *IDOL Speech Server Administration Guide*.

Transcode Audio

In most cases you do not need to transcode audio before sending it to IDOL Speech Server.

TIP:
Transcoding is necessary only when you set `IdolSpeechUseStreaming=TRUE`, which is **not** recommended.

In the following configuration, audio is streamed to the IDOL Speech Server, because the parameter `IdolSpeechUseStreaming` is `TRUE`. The configuration therefore includes the parameters `TranscodeServerHost` and `TranscodeServerPort` so that CFS sends the audio to a Transcode Server before it is sent to Speech Server. Although the audio is streamed to Speech Server, the shared folder specified by the `SharedPath` parameter is required so that CFS and the Transcode Server can exchange data.

```
[ImportTasks]
Pre0=Lua:scripts/IdolSpeech.lua

[IdolSpeechSettings]
IdolSpeechServers=server1:15000,server2:15000
IdolSpeechUseStreaming=TRUE
TranscodeServerHost=server3
```

```
TranscodeServerPort=30000
SharedPath=\\server\SharedPath
IdolSpeechLanguage=ENUK
```

If you are streaming audio to Speech Server but know that files are already in an acceptable format, you can configure CFS to bypass the transcoding step of the `IdolSpeech` task.

To bypass transcoding

- To bypass transcoding for all documents, do *not* set the `TranscodeServerHost` or `TranscodeServerPort` configuration parameters when you configure the `IdolSpeech` task.
- To bypass transcoding for a single document, add the field `AUTN_FORMAT_CORRECT_FOR_TRANSCRIPTION` to the document. The field can have any value. CFS does not send these files to the Transcode Server.

Speech-To-Text Results

When you run speech-to-text, CFS adds a transcription of the speech to the document content (the `DRECONTENT` field).

CFS can also add the start time, duration, and confidence score for each detected word, sentence boundary, and period of silence to the document metadata:

- To add start times and durations to the document metadata, set the parameter `AddTimingsToMetadata=TRUE`.
- To add confidence scores to the document metadata, set the parameter `AddConfidenceToMetadata=TRUE`.

If you choose to add information to the document metadata, CFS adds a metadata field named `SpeechToTextWord` for each detected word, sentence boundary, or period of silence.

When you set `AddTimingsToMetadata=TRUE`, the field includes attributes named `start` and `duration`, which describe the start time and duration in the audio:

```
<SpeechToTextWord start="3.1562" duration="0.3568">hello</SpeechToTextWord>
```

When you set `AddConfidenceToMetadata=TRUE`, the field includes an attribute named `confidence`, which describes the confidence score. The confidence score is a value between 0 (zero) and 1. Higher confidence scores indicate greater confidence of a correct result.

```
<SpeechToTextWord confidence="0.9568">hello</SpeechToTextWord>
```

When you set `AddTimingsToMetadata=TRUE` and `AddConfidenceToMetadata=TRUE`, CFS adds fields that include all of these attributes:

```
<SpeechToTextWord start="3.1562" duration="0.3568"
confidence="0.9568">hello</SpeechToTextWord>
```

Fields that represent periods of silence have no value, for example:

```
<SpeechToTextWord start="3.1562" duration="0.3568" confidence="0.9568" />
```

Fields that represent sentence boundaries have a value of "`.`", for example:

```
<SpeechToTextWord start="3.1562" duration="0.3568"
confidence="0.9568">.</SpeechToTextWord>
```

Categorize Documents

Categorization analyzes the concepts that exist in a document and, if those concepts match categories in IDOL Server, adds category information to the document. Categorizing documents is useful because you can alert IDOL users to new content that matches their interests, help them find information through taxonomies, and help them to identify similar documents.

To use categorization, you must have created and trained categories in IDOL Server. CFS queries IDOL by sending the `CategorySuggestFromText` action for each document, and IDOL returns information about any categories that match. If a document does not match any of the categories in IDOL Server, the document is not categorized. For information about how to create and train categories, refer to the *IDOL Server Administration Guide*.

To categorize documents

1. Stop CFS.
2. Open the CFS configuration file.
3. Create an import task to run the `CategorySuggestFromText` Lua script that is supplied with CFS. For example:

   ```
   [ImportTasks]
   Post0=Lua:./scripts/CategorySuggestFromText.lua
   ```

4. Open the script in a text editor.
5. Modify the variables in the script so that the script sends actions to your IDOL Server:

Line	Variable name	Value
178	`idolCategorizeHost`	The host name or IP address of your IDOL Server.
179	`idolCategorizePort`	The ACI port of your IDOL Server. The `port` argument in the function `send_aci_action` expects a number, so do not surround the port number with quotation marks.
184	`timeoutMilliseconds`	The amount of time, in milliseconds, that CFS waits for a response from your IDOL Server. If CFS does not receive a response within this time limit and the number of retries is reached, the document is not categorized. You should not need to modify the default value, which is 60 seconds.
185	`retries`	The number of times that CFS retries a request to your IDOL Server, if the first attempt is not successful.
186-192	`sslParameters`	A table of SSL parameters for connecting to your IDOL Server. For more information about the SSL parameters that you can set, refer to the *Connector Framework Server Reference*.

For example:

```
local idolCategorizeHost = "10.0.0.1"
local idolCategorizePort = 9000

...

local timeoutMilliseconds = 30000
local retries = 3
local sslParameters =
   {
       SSLMethod = "SSLV23",
       --SSLCertificate = "host1.crt",
       --SSLPrivateKey = "host1.key",
       --SSLCACertificate = "trusted.crt"
   }
```

6. Save and close the script.

Customize the Query

The `CategorySuggestFromText` Lua script sends an entire document (metadata and content) to IDOL for categorization. The document is converted to a string using the `to_idx` method and then passed to the `QueryText` parameter of the `CategorySuggestFromText` action:

```
local categorySuggestFromTextParameters = { QueryText = document:to_idx() }

...

local output = send_aci_action(
               idolCategorizeHost,
               idolCategorizePort,
               "categorysuggestfromtext",
               categorySuggestFromTextParameters,
               timeoutMilliseconds,
               retries,
               sslParameters
               )
```

You can modify the script to categorize the document based on a specific field. For example, to use only the document content:

```
local categorySuggestFromTextParameters = {
                      QueryText = document:getContent()
                      }
```

Alternatively, to use the value of a single document field:

```
local categorySuggestFromTextParameters = {
                      QueryText = document:getFieldValue("MyFieldName")
                      }
```

You can also add additional parameters to the action. For example, the `CategorySuggestFromText` Lua script does not limit the number of categories that are added to the document. To add only the most relevant category to a document, add the `CategorySuggestFromText` action parameter `NumResults=1` by modifying the script as follows:

```
local categorySuggestFromTextParameters = {
                      QueryText = document:getContent(),
                      NumResults = 1
                      }
```

For more information about the `CategorySuggestFromText` action and the parameters that it supports, refer to the *IDOL Server Reference*.

Customize the Output

The `CategorySuggestFromText` Lua script creates the following document fields by default:

Field name	Value
`category_title`	The name of the category.
`category_id`	The ID of the category in IDOL Server.
`category_reference`	The `DREREFERENCE` of the category, stored as a document in the Agentstore.

The script adds one value to each field for each category that matches the document. For example:

```
#DREFIELD category_id="200"
#DREFIELD category_id="100"
#DREFIELD category_reference="200"
#DREFIELD category_reference="100"
#DREFIELD category_title="Science"
#DREFIELD category_title="BusinessNews"
```

To modify how the information is added to the document, customize the Lua script. For example, to change the names of the fields, modify the first argument of the `addField` method on lines 211 to 213:

```
document:addField("category_name", category["title"])
document:addField("category_ref", category["reference"])
document:addField("category_id", category["id"])
```

To add only the category names, remove lines 212 and 213:

```
document:addField("category_title", category["title"])
-- document:addField("category_reference", category["reference"])
-- document:addField("category_id", category["id"])
```

To add all of the category information to a single field, using subfields, you could modify the script as follows (replacing lines 207-219):

```
if(suggestWasSuccessful) then
   local suggestedCategories = parseCategories(output)

   document:addField("category", "category information")
   local field = document:getField("category")
   for i, category in ipairs(suggestedCategories) do
       field:addField("title",category["title"])
       field:addField("reference",category["reference"])
       field:addField("id",category["id"])
   end

   document:setFieldValue("result", output)

   return true
end
```

To add all of the category names to a single field as a comma-separated list, you could modify the script as follows (replacing lines 207-219):

```
if(suggestWasSuccessful) then
   local suggestedCategories = parseCategories(output)

   local names=""
   for i, category in ipairs(suggestedCategories) do
       if i==1 then
          names = category["title"]
       else
          names = names .. "," .. category["title"]
       end
   end

   if names~="" then
      document:addField("category_names_CSV", names)
   end

   document:setFieldValue("result", output)

   return true
end
```

Run Eduction

Eduction identifies and extracts entities from text, based on a pattern that you define. An *entity* is a word, phrase, or block of information. A *pattern* might be a dictionary, for example a list of people or places. Alternatively, the pattern can describe what the entity looks like without having to list it explicitly, for example a regular expression that describes an address or telephone number. After entities are extracted the text is written to the document fields that you specify. For more information about Eduction, refer to the *IDOL Eduction User Guide*.

You can run Eduction on document fields using the `Eduction` task.

NOTE:
To use the `Eduction` task you must have a license for Eduction and the relevant grammar files,

and specify the host name and ACI port of your License Server in the CFS configuration file.

You can run the `Eduction` task using the `Post` parameter. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=Eduction:EductionSettings

[EductionSettings]
ResourceFiles=C:\MyGrammar\gram1.ecr
SearchFields=DRECONTENT
Entity0=edk_common_entities/postal_address
EntityField0=SHIPPING_ADDRESS
```

Redact Documents

You can use the `Eduction` task to redact information in documents.

To enable redaction, set the configuration parameter `RedactedOutput=True`. If you want to specify the value or characters that replace the redacted text, use the configuration parameter `RedactionOutputString` or `RedactionReplacementCharacter`.

For example, the following configuration redacts addresses contained in a document's `DRECONTENT` or `ADDRESS` fields:

```
[ImportTasks]
Post0=Eduction:EductionSettings

[EductionSettings]
ResourceFiles=C:\Autonomy\IDOLServer\Eduction\address_gb.ecr
SearchFields=DRECONTENT,ADDRESS
RedactedOutput=True
```

The fields specified by `SearchFields` are not modified. CFS places the redacted text in fields with a `_REDACTED` suffix. For example:

```
#DREFIELD ADDRESS="Cambridge Business Park, Cowley Road, Cambridge, CB4 0WZ"
#DREFIELD ADDRESS_REDACTED="[redacted]"
```

The `Eduction` task also adds the value, offset, and score for any matched entities to the document. For example:

```
#DREFIELD /offset="298"
#DREFIELD /score="1"
#DREFIELD /value="Cambridge Business Park, Cowley Road, Cambridge, CB4 0WZ"
```

Lua Post Processing

An *Eduction Lua Post Processing task* runs a Lua script that modifies the output from the Eduction module. For example, you might want to increase the score for a match if it is found near similar matches.

> **NOTE:**
> The Lua script is run by the Eduction module, not by CFS. The Eduction module expects the script to start with `function processmatch (edkmatch)`. You cannot modify the document being processed by CFS, or use the Lua methods that are available to CFS Lua scripts. For information about the Lua methods that are available in the Eduction module, refer to the *Eduction User Guide*.

To create an Eduction Lua Post Processing task, set the `PostProcessingTaskN` parameter. This specifies the name of a section in the CFS configuration file that contains parameters to configure the task. For example:

```
[ImportTasks]
Post0=Eduction:EductionSettings

[EductionSettings]
ResourceFiles=C:\MyGrammar\gram1.ecr
SearchFields=DRECONTENT
Entity0=edk_common_entities/postal_address
EntityField0=SHIPPING_ADDRESS
PostProcessingTask0=EductionLuaPostProcessing

[EductionLuaPostProcessing]
Script=scripts/eduction_post_process.lua
ProcessEnMasse=False
```

The Eduction Lua module will call `function processmatch (edkmatch)`. For example:

```
function processmatch(edkmatch)
   if edkmatch then
      local text = edkmatch:getOutputText()
      -- modify the match
      edkmatch:setOutputText(text)
      return true
   end

   return false -- return false to drop the match
end
```

The `edkmatch` argument represents a single eduction match, or the complete set of matches if you set the configuration parameter `ProcessEnMasse` to `true`.

If the `processmatch` function returns `true`, the match is returned to CFS. If the function returns `false`, the match is discarded.

For more information about writing Eduction post-processing scripts, and information about the Lua methods that you can use, refer to the *Eduction User Guide*.

Process HTML

Connectors, including the IDOL Web Connector, can send documents to CFS that have associated HTML files.

CFS can send the HTML files to KeyView, which discards the HTML markup and extracts the text contained in the file. However, HTML pages often contain irrelevant content such as invalid HTML, headers, sidebars, advertisements, and scripts. This text does not contain any useful information and could pollute the IDOL index, degrading performance. KeyView does not remove this irrelevant content, so Connector Framework Server provides features to process HTML files.

- **HTML processing with WKOOP**. CFS can use an embedded browser (WKOOP) to process HTML in a similar way to the IDOL Web Connector. There are many reasons to use WKOOP over other methods of processing HTML:
  - The browser allows scripts to run before the page is processed, so CFS can extract content and links that are added by JavaScript.
  - Links are resolved before a document is ingested, so that indexed documents contain absolute URLs.
  - You can remove unwanted content using the automatic clipping algorithm, or by selecting parts of the page with CSS selectors.
  - You can extract metadata or divide pages into multiple documents using CSS selectors rather than regular expressions.

  > **NOTE:**
  > To use WKOOP you must also install the IDOL Web Connector, because WKOOP is not provided with CFS. You must install a version of WKOOP that is the same as, or later than, the version of CFS that you are using.

- **HTMLExtraction**. HTML extraction extracts the useful information from the page and discards the irrelevant content. It automatically determines which content is relevant, so there are no configuration parameters for customizing this operation. If HTML extraction does not produce good results for your use case, you might want to use the clipping features provided by WKOOP, instead.

HTML Processing with WKOOP

The `WKOOPHtmlExtraction` task processes an HTML file that is associated with a document. It extracts links and metadata and adds these to the document in a metadata field named `HTML_PROCESSING`. The task appends a page to the document content that contains the plain text extracted from the HTML source. It also sets the field `AUTN_NO_FILTER`, to prevent the document being processed by KeyView.

This section describes how to configure HTML processing with WKOOP.

You can configure WKOOP HTML extraction as a pre-import task (`Pre0` in the following example). The `Pre0` parameter also specifies the name of a section that contains the settings for the task. In the following example the section is named `HtmlProcessingSettings`.

```
[ImportTasks]
Pre0=WKOOPHtmlExtraction:HtmlProcessingSettings

[HtmlProcessingSettings]
WKOOPPath=F:\IDOL\WebConnector\WKOOP.exe
ProxyHost=proxy.domain.com
ProxyPort=8080
SSLMethod=NEGOTIATE
```

```
ExtractLinks=TRUE
ResolveLinks=TRUE
Url=https://www.example.com/
```

The `WKOOPPath` parameter specifies the path to WKOOP. WKOOP is not included with CFS, so you must install the IDOL Web Connector and specify the path to the WKOOP executable file. You must install a version of WKOOP that is the same as, or later than, the version of CFS that you are using.

If you are running CFS on a machine that is behind a proxy server, set the `ProxyHost` and `ProxyPort` parameters to specify the proxy server to use to access the web. The `SSLMethod` parameter specifies the version of SSL or TLS to use when connecting to the web site, and is necessary to retrieve resources over HTTPS. Setting this parameter to `NEGOTIATE` uses the latest version that is supported by both CFS and the web server.

The `ExtractLinks` parameter accepts a Boolean value that specifies whether to extract links from HTML pages and add the links to the document metadata. When `ResolveLinks=TRUE` the links are resolved so that indexed documents contain absolute URLs. The `Url` parameter specifies the source URL so that links can be resolved. You do not need to specify the exact URL of the page being processed, as long as all URLs in the document being processed are relative to the web server.

For a full list of configuration parameters that you can use to configure WKOOP HTML extraction, refer to the *Connector Framework Server Reference*.

Remove Irrelevant Content

To remove irrelevant content from HTML pages using the automatic clipping algorithm, add the parameter `Clipped=TRUE` to your task configuration. CFS decides which parts of the page to keep and which to discard.

The automatic clipping algorithm has been designed to work with many different pages, but this means that automatic clipping might not give the best results for every page. Alternatively, you can use CSS selectors to choose which parts of the page to keep and which to discard. To clip pages with CSS selectors, add `Clipped=TRUE` to your task configuration, and then set `ClipPageUsingCssSelect` to specify the parts of the page to keep and `ClipPageUsingCssUnselect` to specify the parts of the page to remove. These parameters accept standard CSS2 selectors.

You can also remove scripts and hidden content from the HTML page:

- Remove all scripts from the HTML page by setting `RemoveScripts=TRUE`.
- Remove "noframes" content by setting `RemoveNoframes=TRUE`. When web developers use frames they might include content in a `<noframes></noframes>` element, for web browsers that do not support frames. This content might duplicate content elsewhere in the HTML page or simply contain a message that the browser does not support frames.

Extract Metadata

This section demonstrates how to extract metadata from an HTML page and add it to a document field.

Consider the following HTML:

```
<h1>This is a title</h1>
<h2>This is a sub-title</h2>
```

```
<p class="important">This is <strong>important</strong> text</p>
```

From this HTML you could extract all of the headings and add them to a metadata field named `heading`. You could also extract the important text and add that to a separate document field.

The configuration parameters `MetadataSelector` and `MetadataFieldName` select the information to extract and provide the name of the destination document field. These parameters must be set in numbered pairs (so that each `MetadataSelector` parameter has a matching `MetadataFieldName`). The `MetadataSelector` parameter accepts standard CSS2 selectors.

The following configuration would extract the information described above:

```
MetadataSelector0=h1,h2,h3
MetadataFieldName0=heading
MetadataSelector1=p.important
MetadataFieldName1=important_paragraph
MetadataSelectorExtractPlainText=TRUE
```

The parameter `MetadataSelectorExtractPlainText` specifies whether to extract as plain text (removing HTML markup, for example).

The configuration above would produce the following metadata fields:

```
#DREFIELD heading="This is a title"
#DREFIELD heading="This is a sub-title"
#DREFIELD important_paragraph="This is important text"
```

Split Web Pages into Multiple Documents

You might want to split pages into multiple documents. For example, if you ingest pages from a discussion board you might want to ingest one document for each message on the page.

Connector Framework Server can create documents for sections of a Web page identified using CSS selectors. CFS creates a child document for each section of the page that is identified. Metadata fields (named `CHILD_DOCUMENT`) are added to the parent document, to refer to the child documents.

To split pages into multiple documents, add the following parameters to your `WKOOPHtmlExtraction` task:

`ChildDocumentSelector`	A CSS2 selector that identifies the root element of each child document in the page source.
`ChildReferenceSelector`	(Optional) An element in the child document that contains a value to use as the document reference. The value you extract should be unique for each child document, because it is used as part of the `DREREFERENCE` field in the child document. If you do not set this parameter, the connector uses a GUID. Specify the element using a CSS2 selector, relative to the element identified by `ChildDocumentSelector`.
`ChildMetadataSelector`	(Optional) A list of elements in the child document that contain metadata. The metadata in these elements are extracted and added to the metadata fields of child documents. Specify the elements as a list of CSS2 selectors, relative to the element identified by

	`ChildDocumentSelector`. To specify the name(s) of the document field(s) to contain the extracted information, set the configuration parameter `ChildMetadataFieldName`. Both parameters must have the same number of values.
`ChildMetadataFieldName`	(Optional) The names to use for document fields (in child documents) that contain information extracted using the parameter `ChildMetadataSelector`. This parameter must have the same number of values as `ChildMetadataSelector`.

For example, consider the following example page which represents messages on a page of a discussion board:

```
<html>
    <head>
        <title>Example Page</title>
        <meta charset="utf-8">
    </head>
    <body>
        <div>
            <h1>Example Page</h1>
            <div class="content">
                <p>content</p>
            </div>
            <div class="message">
                <h1>Message 1</h1>
                <p class="meta">some metadata</p>
                <p>some content</p>
            </div>
            <div class="message">
                <h1>Message 2</h1>
                <p class="meta">some metadata</p>
                <p>some content</p>
            </div>
            ...
        </div>
    </body>
</html>
```

To create separate documents for the messages contained on this page, you could use the following configuration:

```
[MyTask]
...
ChildDocumentSelector=div.message
ChildReferenceSelector=h1
ChildMetadataFieldName0=my_metadata
ChildMetadataSelector0=p.meta
```

This example would produce the following child document (and a similar document for the second message):

```
#DREREFERENCE <current_document_reference>:<child_reference>
#DREFIELD my_metadata="some metadata"
...
#DRECONTENT
Message 1
some metadata
some content
...
```

The value of the `DREREFERENCE` field is constructed from the reference of the original document and the value of the element identified by the `ChildReferenceSelector` configuration parameter. If you don't set this configuration parameter or the element is not found, CFS uses a GUID instead.

CFS adds the reference of the original document to the fields `DREPARENTREFERENCE` and `DREROOTPARENTREFERENCE`. It also adds an `HTML_PROCESSING` metadata field that contains any metadata and links that are extracted from the child document.

The `DRECONTENT` field is populated with text extracted from the HTML elements that you identified as belonging to the child document.

Connector Framework Server automatically adds fields to the parent document, named `CHILD_DOCUMENT`, that contain the references of associated child documents.

HTML Extraction

HTML pages often contain irrelevant content such as invalid HTML, headers, sidebars, advertisements, and scripts. CFS can extract the useful information from the page and discard the irrelevant content.

To extract the useful information from an HTML page, use the `HtmlExtraction` import task. This task works only on HTML files and ignores other file types.

CFS reads the HTML document, and discards data such as invalid HTML, headers, sidebars, advertisements, and scripts. In the remaining content, CFS then extracts blocks of text that contain a large number of stopwords and a low proportion of links. This text is likely to be the most important content. Because CFS automatically determines which content is relevant, there are no configuration parameters for customizing this task.

Micro Focus recommends that you configure the `HtmlExtraction` task as a *Pre* import task. For example:

```
[ImportTasks]
Pre0=HtmlExtraction
```

After extracting the useful information, the HTML Extraction task sets the document field AUTN_NO_FILTER, so that the HTML file is not processed by KeyView.

Extract Metadata from Files

The `ExtractMetadata` task extracts metadata from the file associated with a document. This task extracts a subset of the metadata obtained by standard KeyView filtering. It is faster than standard KeyView filtering and does not extract the file content.

> **TIP:**
> When documents are ingested, CFS automatically extracts metadata. Do not use this task unless you have set the fields `AUTN_NO_FILTER` and `AUTN_NO_EXTRACT` on a document and want to extract basic metadata only.

The `ExtractMetadata` task is configured as a *Pre* task. Specify the name of the section that contains settings for the task. For example:

```
[ExtractMetadata]
Pre0=Lua:scripts/nofilter.lua
Pre1=ExtractMetadata:ExtractMetadataSettings

[ExtractMetadataSettings]
FieldnamePrefix=FIELD_
ReservedFieldnames=Reserved1,Reserved2
```

The `Pre0` task runs a Lua script that adds the fields `AUTN_NO_FILTER` and `AUTN_NO_EXTRACT` to documents. Adding these fields prevents KeyView from filtering the documents and extracting subfiles.

The `Pre1` task runs the `ExtractMetadata` task using the settings contained in the `[ExtractMetadataSettings]` section of the CFS configuration file.

The `FieldnamePrefix` parameter specifies a prefix for the names of the metadata fields that are added to the document. The `ReservedFieldnames` parameter specifies a comma-separated list of field names that the task must not use. If the task needs to add a metadata field with one of the specified names, it prefixes the name with an underscore. For example, with the settings specified above, the task would not add a field named `FIELD_Reserved1`. Instead, the task would add `_FIELD_Reserved1`.

Import Content Into a Document

The `ImportFile` task imports a file and adds its content to the document being processed. CFS does not extract sub files from the file.

The `ImportFile` task can be configured as a *Pre* or *Post* task. When you create the task, specify the name of a document field that contains the path or URL of the file to import, for example:

```
Pre0=ImportFile:fieldname
Post0=ImportFile:fieldname
```

where **`fieldname`** is the name of the document field.

Alternatively, specify the name of a section in the configuration file that contains the settings for the task:

```
[ImportTasks]
Post0=ImportFile:MySettings

[MySettings]
Fieldname=field_containing_file_path_or_url
ProxyHost=10.0.0.1
ProxyPort=8080
SSLMethod=TLSV1
```

If the field contains a URL, CFS downloads the file and adds its content to the document.

Reject Invalid Documents

You can configure CFS to reject documents based on several criteria.

When documents are rejected, they are not processed by further tasks. You can index rejected documents or discard them:

- To index the documents into one or more indexes, such as an IDOL Error Server, set the parameters `OnErrorIndexerSections` and `IndexDatabase`. `OnErrorIndexerSections` specifies a list of configuration file sections to use to index a document. These sections must contain indexing parameters, such as the host name and ACI port of your IDOL Server. `IndexDatabase` specifies the name of the IDOL database into which the rejected documents are indexed. Before indexing a document, CFS writes the name of the filter that caused the document to be rejected to a field named `MATCHEDFILEFILTERS`.
- If you do not specify any indexing details, the documents are discarded. CFS writes a message to the import log showing that the document was rejected, and showing which filter caused the rejection.

Reject Documents with Binary Content

The `BinaryFileFilter` task rejects any documents that have been filtered as binary. This can occur when KeyView filtering fails, for example due to corrupt files.

When CFS detects a non-UTF8 character, it replaces the character with a hexadecimal character code. The `BinaryFileFilter` task detects these character codes and rejects documents where the proportion exceeds the limit set by the `ThresholdPercent` parameter.

The `BinaryFileFilter` task can be configured as a *Post* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=BinaryFileFilter:BinaryFileFilterSettings

[BinaryFileFilterSettings]
ThresholdPercent=10
```

```
OnErrorIndexerSections=IdolErrorServer
IndexDatabase=IdolErrorReview
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Reject Documents with Import Errors

The `ImportErrorFilter` task rejects any documents for which errors have occurred. Errors can occur during KeyView filtering or during *pre* and *post* import tasks.

The `ImportErrorFilter` task can be configured as a *Post* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=ImportErrorFilter:ImportErrorFilterSettings

[ImportErrorFilterSettings]
OnErrorIndexerSections=IdolErrorServer
IndexDatabase=IdolErrorReview
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Reject Documents with Symbolic Content

The `SymbolicContentFilter` task calculates the proportion of symbolic characters in a document. If the proportion of symbolic characters in the document content exceeds the limit specified by the `MaxSymbolicCharactersPercent` parameter, the document is rejected.

Symbolic characters are defined as any character between U+2000 and U+2FFF.

The `SymbolicContentFilter` task can be configured as a *Post* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=SymbolicContentFilter:SymbolicContentFilterSettings

[SymbolicContentFilterSettings]
MaxSymbolicCharactersPercent=8
OnErrorIndexerSections=IdolErrorServer
IndexDatabase=IdolErrorReview
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Reject Documents by Word Length

The `WordLengthFilter` task calculates the average length of words in a document. If the average length of words in the document content (`DRECONTENT`) falls outside the limits specified by the

`MinimumAverage` or `MaximumAverage` parameters, the document is rejected.

The `WordLengthFilter` task can be configured as a *Post* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=WordLengthFilter:WordLengthFilterSettings

[WordLengthFilterSettings]
MinimumAverage=3.0
MaximumAverage=10.0
OnErrorIndexerSections=IdolErrorServer
IndexDatabase=IdolErrorReview
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Reject All Invalid Documents

The `BadFilesFilter` task rejects all documents that are considered to be invalid:

- Documents that have binary content.
- Documents for which import errors have occurred.
- Documents that have too high a proportion of symbolic content.
- Documents where the average word length is too long or too short.

`BadFilesFilter` must be configured as a *Post* task.

`BadFilesFilter` reads configuration parameters from the section of the configuration file that you specify in the `Post` parameter. In this section you can set parameters for each filter. In the example below, two parameters have been set to configure the word length filter:

```
[ImportTasks]
Post0=BadFilesFilter:BadFilesFilterSettings

[BadFilesFilterSettings]
MinimumAverage=3.0
MaximumAverage=10.0
OnErrorIndexerSections=IdolErrorServer
IndexDatabase=IdolErrorReview
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference.*

Split Document Content into Sections

Dividing the content of long documents into sections can result in more relevant search results, because IDOL Server can return a specific part of a document in response to a query.

To divide document content into sections, use the `Sectioner` task.

The `Sectioner` import task must be configured as a *Post* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Post0=Sectioner:Sectioning

[Sectioning]
SectionerMaxBytes=3000
SectionerMinBytes=1500
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Split Files into Multiple Documents

Sometimes you might retrieve files from a repository that you would prefer to ingest as multiple documents.

You can use the `TextToDocs` task to split a file containing text (for example an HTML file or XML file) into multiple documents. To divide a file, you specify regular expressions that match the relevant parts of the document. The task creates a main document and one or more child documents, which can all have metadata and content. When you run `TextToDocs` on a document, the original document is discarded. The documents created by `TextToDocs` are metadata-only documents, which means that they do not have an associated file and are not filtered by KeyView.

The `TextToDocs` task should be configured as a *Pre* task. The parameters that are passed to the task are specified in a named section of the configuration file. For example:

```
[ImportTasks]
Pre0=TextToDocs:MyTextToDocs

[MyTextToDocs]
...
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

The `TextToDocs` task expects documents to use UTF-8 character encoding. If your documents are not encoded in UTF-8 you can use the configuration parameter `SourceEncoding` to specify the character set encoding of the source documents, so that they can be converted to UTF-8. If conversion fails, the original encoding is used and CFS adds an error message to the `ImportErrorCode` and `ImportErrorDescription` document fields.

Example

The following HTML is an example file that you might want to ingest as separate documents. There are clear sections which could represent different topics:

```
<html>
  <body>
    <p class="main">Main content</p>
    <div class="section">
      <h1>First document</h1>
      <p class="metadata">Extract as metadata</p>
      <p>Some text</p>
    </div>

    <div class="section">
       <h1>Second document</h1>
       <p class="metadata">Extract as metadata</p>
       <p>Some text</p>
    </div>

    <div class="section">
       <h1>Third document</h1>
       <p class="metadata">Extract as metadata</p>
       <p>Some text</p>
    </div>

  </body>
</html>
```

You might want to split this file into a main document and three child documents, one of which might look like this:

```
#DREREFERENCE C:\MyFiles\TextToDocs\textToDocs.html:0
#DREDBNAME FileSystem
#DREFIELD MyMetadataField="Extract as metadata"
#DRECONTENT
First document
Some text

#DREENDDOC
```

To do this, you could use the following configuration:

```
[ImportTasks]
Pre0=TextToDocs:MyTextToDocs

[MyTextToDocs]
FilenameMatchesRegex0=.*\.html

MainRangeRegex0=<html>(.*)</html>
MainContentRegex0=<p class="main">(.*?)</p>

ChildrenRangeRegex0=<html>(.*)</html>
ChildRangeRegex=<div class="section">(.*?)</div>
ChildContentRegex0=<h1>(.*?)</h1>
ChildContentRegex1=<p>(.*?)</p>
ChildFieldName0=MyMetadataField
```

```
ChildFieldRegex0=<p class="metadata">(.*?)</p>
ChildInheritFields=DREDBNAME
```

In this example, the `FilenameMatchesRegex` parameter has been set to process only those files that have the extension `.html`.

The `MainContentRegex` parameter identifies parts of the original document to add to the `DRECONTENT` field of the main document.

The `ChildRangeRegex` parameter identifies the parts of the original document that should become child documents. The sub-match `(.*?)` matches all of the content between a `<div class="section">` tag and a `</div>` tag. When this regular expression is applied to the example document above, there are three matches and therefore three child documents are created. It is important to make the regular expression lazy, because otherwise it would match everything between the first `<div class="section">` and the final `</div>`, resulting in a single child document.

The `ChildContentRegex` parameter identifies the content to add to the `DRECONTENT` field of a child document. In this example two regular expressions are used to extract content. The `ChildFieldName` and `ChildFieldRegex` parameters populate metadata fields. In this example a single field named `MyMetadataField` is created.

Setting the parameter `ChildInheritFields=DREDBNAME` specifies that the child documents inherit the field `DREDBNAME` from the original document. If you are indexing documents into IDOL Server it is important to set this parameter, because (depending on how your system is configured) documents without a `DREDBNAME` field might not be indexed.

Write Documents to Disk

CFS can write documents to disk in several formats. You might want to write documents to disk for the following reasons:

- so that you can see the data that is being indexed into IDOL Server, Haven OnDemand, or Vertica. You can then set up further processing tasks to manipulate and enrich the data.
- so that you can debug your Lua scripts or other processing tasks.
- so that you can export the data from documents to other systems.

Write Documents to Disk in IDX Format

To write documents to disk in IDX format, configure an `IdxWriter` processing task by setting the `Pre`, `Post`, `Update`, or `Delete` configuration parameter.

To run the `IdxWriter` task with default settings, use the `Pre`, `Post`, `Update`, or `Delete` parameter to specify the file name for the IDX file:

```
[ImportTasks]
Pre0=IdxWriter:pre.idx
Post0=IdxWriter:post.idx
```

Alternatively, set the parameter to `IdxWriter`, followed by a colon (`:`), followed by the name of the section in the configuration file that contains custom settings for the task. For example:

```
[ImportTasks]
Pre0=IdxWriter:PreIDX
Post0=IdxWriter:PostIDX

[PreIDX]
IdxWriterFilename=pre.idx
IdxWriterMaxSizeKBs=100
IdxWriterArchiveDirectory=./IDXArchive

[PostIDX]
IdxWriterFilename=post.idx
IdxWriterMaxSizeKBs=100
IdxWriterArchiveDirectory=./IDXArchive
```

For information about the configuration parameters you can use to configure this task, refer to the *Connector Framework Server Reference*.

Write Documents to Disk in XML Format

To write documents to disk in XML format, configure an `XmlWriter` processing task by setting the `Pre`, `Post`, `Update`, or `Delete` configuration parameter.

When you create the `XmlWriter` task, specify the file name of the XML file. For example:

```
[ImportTasks]
Pre0=XmlWriter:C:\ConnectorFrameworkServer\pre.xml
Post0=XmlWriter:C:\ConnectorFrameworkServer\post.xml
```

Write Documents to Disk in JSON Format

To write documents to disk in JSON format, configure a `JsonWriter` processing task by setting the `Pre`, `Post`, `Update`, or `Delete` configuration parameter.

To configure the task with default settings, specify the file name for the output file:

```
[ImportTasks]
Pre0=JsonWriter:pre.jsn
Post0=JsonWriter:post.jsn
```

Alternatively, set the parameter to `JsonWriter`, followed by a colon (`:`), followed by the name of the section in the configuration file that contains custom settings for the task. For example:

```
[ImportTasks]
Post0=JsonWriter:PostJsonWriting

[PostJsonWriting]
JsonWriterFilename=post.jsn
```

```
JsonWriterMaxSizeKBs=1000
JsonWriterArchiveDirectory=./JSONarchive
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Write Documents to Disk in CSV Format

The `CsvWriter` task writes document metadata and content to a comma-separated values (CSV) file. This allows you to export the data to other systems.

The task always writes the document reference (`DREREFERENCE`) and content (`DRECONTENT`) fields, and you can choose the other fields that you want to include. The task writes the field names, followed by one line of values for each document that is ingested.

The `CsvWriter` task can be configured as a *Pre*, *Post*, *Update* or *Delete* task.

To run the task with default settings, specify the file name for the output file:

```
[ImportTasks]
Post0=CsvWriter:MyTask.csv
```

Alternatively, specify the name of a section in the configuration file that contains the settings for the task:

```
[ImportTasks]
Post0=CsvWriter:CsvWriting

[CsvWriting]
CsvWriterFilename=MyTask.csv
CsvWriterMaxSizeKBs=1000
CsvWriterArchiveDirectory=./CSVarchive
CsvWriterFieldNames0=A_FIELD
CsvWriterFieldNames1=A_FIELD/subfield
CsvWriterFieldNames2=A_FIELD/@attribute
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Write Documents to Disk as SQL INSERT Statements

The `SqlWriter` task writes document metadata and content to a file in the form of SQL “INSERT” statements. You can use the SQL to insert the data from the documents into a database.

The task always writes the document reference (`DREREFERENCE`) and content (`DRECONTENT`) fields, and you can choose the other fields that you want to include. The task writes one INSERT statement for each document that is processed.

The `SqlWriter` task can be configured as a *Pre*, *Post*, *Update* or *Delete* task.

To configure the task, specify the name of a section that contains the settings, for example:

```
[ImportTasks]
Post0=SqlWriter:SqlWriting

[SqlWriting]
SqlWriterFileName=MyTask.sql
SqlWriterTableName=table
SqlWriterDreReferenceColumnName=REFERENCE
SqlWriterDreContentColumnName=CONTENT

SqlWriterFieldNames0=MODIFIED_DATE
SqlWriterColumnNames0=DATE
SqlWriterDataTypes0=DATE_TIME

SqlWriterUseNullForMissingFields=true

SqlWriterDateFormats0=DD/MM/YYYY
SqlWriterDateFormats1=YYYY/MM/DD

SqlWriterMaxSizeKBs=1024
SqlWriterArchiveDirectory=./SQLarchive
```

For information about the parameters that you can use to configure this task, refer to the *Connector Framework Server Reference*.

Standardize Document Fields

The documents created by your connectors might not have consistent field names. For example, documents created by the File System Connector can have a field named `FILEOWNER`. Documents created by the Documentum Connector can have a field named `owner_name`. Both of these fields store the name of the person who owns a file.

You might want to rename document fields so that documents use the same field names to store the same type of information. CFS includes the `standardizer` task to do this.

You can configure the `Standardizer` task as a *Pre* or *Post* task. For example:

```
[ImportTasks]
Post0=Standardizer
```

To use the `Standardizer` task, you must set the `EnableFieldNameStandardization` and `FieldNameDictionaryPath` configuration parameters in the `[ImportService]` section of the CFS configuration file. For more information about these parameters, refer to the *Connector Framework Server Reference*.

Customize Field Standardization

Field standardization modifies documents so that they have a consistent structure and consistent field names. You can use field standardization so that documents indexed into IDOL through different connectors use the same fields to store the same type of information. Field standardization only modifies fields that are specified in a dictionary, which is defined in XML format. A standard dictionary, named `dictionary.xml`, is supplied in the CFS installation folder.

In most cases you should not need to modify the standard dictionary, but you can modify it to suit your requirements or create dictionaries for different purposes. By modifying the dictionary, you can configure CFS to apply rules that modify documents before they are ingested. For example, you can move fields, delete fields, or change the format of field values.

The following examples demonstrate how to perform some operations with field standardization.

The following rule renames the field `Author` to `DOCUMENT_METADATA_AUTHOR_STRING`. This rule applies to all components that run field standardization and applies to all documents.

```
<FieldStandardization>
    <Field name="Author">
        <Move name="DOCUMENT_METADATA_AUTHOR_STRING"/>
    </Field>
</FieldStandardization>
```

The following rule demonstrates how to use the `Delete` operation. This rule instructs CFS to remove the field `KeyviewVersion` from all documents. The `Product` element ensures that this rule is run only by CFS.

```
<FieldStandardization>
    <Product key="ConnectorFrameWork">
        <Field name="KeyviewVersion">
            <Delete/>
        </Field>
    </Product>
</FieldStandardization>
```

There are several ways to select fields to process using the `Field` element.

Field element attribute	Description	Example

`name`	Select a field where the field name matches a fixed value.	Select the field `MyField`:
		`<Field name="MyField">`
		`   ...`
		`</Field>`
		Select the field `Subfield`, which is a subfield of `MyField`:
		`<Field name="MyField">`
		`   <Field name="Subfield">`
		`      ...`
		`   </Field>`
		`</Field>`

`path`	Select a field where its path matches a fixed value.	Select the field `Subfield`, which is a subfield of `MyField`. `<Field path="MyField/Subfield">`

		`   ...`
		`</Field>`

`nameRegex`	Select all fields at the current depth where the field name matches a regular expression.	In this case the field name must begin with the word `File`:
		`<Field nameRegex="File.*">`
		`   ...`
		`</Field>`

`pathRegex`	Select all fields where the path of the field matches a regular expression.
	This operation can be inefficient because every metadata field must be checked. If possible, select the fields to process another way.
		This example selects all subfields of `MyField`.
		`<Field pathRegex="MyField/[^/]*">`
		`   ...`
		`</Field>`
		This approach would be more efficient:
		`<Field name="MyField">`
		`   <Field nameRegex=".*">`
		`      ...`
		`   </Field>`
		`</Field>`

You can also limit the fields that are processed based on their value, by using one of the following:

Field element attribute	**Description**	**Example**

`matches`	Process a field if its value matches a fixed value.	Process a field named `MyField`, if its value matches `abc`.
		`<Field name="MyField" matches="abc">`
		`   ...`
		`</Field>`

`matchesRegex`	Process a field if its entire value matches a regular expression.	Process a field named `MyField`, if its value matches one or more digits.
		`<Field name="MyField" matchesRegex="\d+">`
		`   ...`
		`</Field>`

`containsRegex`	Process a field if its value contains a match to a regular expression.	Process a field named `MyField` if its value contains three consecutive digits.
		`<Field name="MyField" containsRegex="\d{3}">`
		`   ...`
		`</Field>`

The following rule deletes every field or subfield where the name of the field or subfield begins with `temp`.

```
<FieldStandardization>
    <Field pathRegex="(.*/)?temp[^/]*">
        <Delete/>
    </Field>
</FieldStandardization>
```

The following rule instructs CFS to rename the field `Author` to `DOCUMENT_METADATA_AUTHOR_STRING`, but only when the document contains a field named `DocumentType` with the value `230` (the KeyView format code for a PDF file).

```
<FieldStandardization>
    <Product key="ConnectorFrameWork">
        <IfField name="DocumentType" matches="230"> <!-- PDF -->
            <Field name="Author">
                <Move name="DOCUMENT_METADATA_AUTHOR_STRING"/>
            </Field>
        </IfField>
    </Product>
</FieldStandardization>
```

TIP:
In this example, the `IfField` element is used to check the value of the `DocumentType` field. The `IfField` element does not change the current position in the document. If you used the `Field` element, field standardization would attempt to find an `Author` field that is a subfield of `DocumentType`, instead of finding the `Author` field at the root of the document.

The following rules demonstrate how to use the `ValueFormat` operation to change the format of dates. The only format that you can convert date values into is the IDOL AUTNDATE format. The first rule transforms the value of a field named `CreatedDate`. The second rule transforms the value of an attribute named `Created`, on a field named `Date`.

```
<FieldStandardization>
    <Field name="CreatedDate">
        <ValueFormat type="autndate" format="YYYY-SHORTMONTH-DD HH:NN:SS"/>
    </Field>
    <Field name="Date">
        <Attribute name="Created">
            <ValueFormat type="autndate" format="YYYY-SHORTMONTH-DD HH:NN:SS"/>
        </Attribute>
    </Field>
</FieldStandardization>
```

As demonstrated by this example, you can select field attributes to process in a similar way to selecting fields.

You must select attributes using either a fixed name or a regular expression:

Select a field attribute by name	`<Attribute name="MyAttribute">`
Select attributes that match a regular expression	`<Attribute nameRegex=".*">`

You can then add a restriction to limit the attributes that are processed:

Process an attribute only if its value matches a fixed value	`<Attribute name="MyAttribute" matches="abc">`
Process an attribute only if its value matches a regular expression	`<Attribute name="MyAttribute" matchesRegex=".*">`
Process an attribute only if its value contains a match to a regular expression	`<Attribute name="MyAttribute" containsRegex="\w+">`

The following rule moves all of the attributes of a field to sub fields, if the parent field has no value. The `id` attribute on the first `Field` element provides a name to a matching field so that it can be referred to by later operations. The `GetName` and `GetValue` operations save the name and value of a selected field or attribute (in this case an attribute) into variables (in this case `$'name'` and `$'value'`) which can be used by later operations. The `AddField` operation uses the variables to add a new field at the selected location (the field identified by `id="parent"`).

```
<FieldStandardization>
    <Field pathRegex=".*" matches="" id="parent">
        <Attribute nameRegex=".*">
            <GetName var="name"/>
            <GetValue var="value"/>
            <Field fieldId="parent">
                <AddField name="$'name'" value="$'value'"/>
            </Field>
            <Delete/>
        </Attribute>
    </Field>
</FieldStandardization>
```

The following rule demonstrates how to move all of the subfields of `UnwantedParentField` to the root of the document, and then delete the field `UnwantedParentField`.

```
<FieldStandardization id="root">
    <Product key="ConnectorFrameWork">
        <Field name="UnwantedParentField">
            <Field nameRegex=".*">
                <Move destId="root"/>
            </Field>
            <Delete/>
        </Field>
    </Product>
</FieldStandardization>
```

Normalize E-mail Addresses

Documents can contain e-mail addresses in many formats, and often the name of the sender or recipient is contained in the same metadata field as their e-mail address.

The `EmailAddressNormalisation` task searches metadata fields for the names and e-mail addresses of e-mail senders and recipients. It then writes the information back to the document in a standard format. For named e-mail addresses (`"Name" <name@domain.com>`), the task separates the name from the address. The task also converts all e-mail addresses to lower-case.

For example, a document might include the following field:

```
<To>"One, Some" <Someone@Somewhere.com>, <user.name@domain.com>, "Else, Someone" <
SomeoneElse@Somewhere.com ></To>
```

The `EmailAddressNormalisation` task reads this information and adds the following fields to the document:

```
<to_email>someone@somewhere.com</to_email>
<to_email>user.name@domain.com</to_email>
<to_email>someoneelse@somewhere.com</to_email>
<to_name>One, Some</to_name>
<to_name/>
<to_name>Else, Someone</to_name>
```

As shown in the previous example, when an e-mail address does not have an associated name, an empty name field is added to the document. This is necessary because the order of the fields in the document is the only way to determine which name belongs with which e-mail address. The first e-mail address is associated with the first name, the second e-mail address with the second name, and so on.

This means that if your source field does not contain any names:

```
<To>Someone@Somewhere.com, SomeoneElse@Somewhere.com</To>
```

The task writes the following fields to the document:

```
<to_email>someone@somewhere.com</to_email>
<to_email>someoneelse@somewhere.com</to_email>
<to_name/>
<to_name/>
```

You can configure `EmailAddressNormalisation` as a *Pre* or *Post* task. For example:

```
[ImportTasks]
Post0=EmailAddressNormalisation:EmailAddressNormalisationSettings

[EmailAddressNormalisationSettings]
FieldNameRegex="To","From","Cc","Bcc"
AddresseeFieldName="to_name","from_name","cc_name","bcc_name"
EmailFieldName="to_email","from_email","cc_email","bcc_email"
```

The `Post0` task runs e-mail address normalisation using the settings in the `[EmailAddressNormalisationSettings]` section. The `FieldNameRegex` parameter specifies a list of regular expressions that identify the fields to process. The `AddresseeFieldName` and `EmailFieldName` parameters specify the names of the fields to add to the document. CFS adds the name of the sender or recipient to the addressee field and their e-mail address to the e-mail field.

Language Detection

CFS can identify the language of a document, and write the name of the language to a document field. A front-end application could use this field to provide a way to filter documents by language. You can also use language detection to reject invalid documents (when a language cannot be detected).

Language detection can be configured as a post-import task. Set the `Post` parameter to `LangDetect` and specify the name of a configuration file section that contains the task settings. For example:

```
[ImportTasks]
Post0=LangDetect:LangDetectSettings

[LangDetectSettings]
LanguageDetectionDirectory=./filters/datafiles/
OutputField=DetectedLanguage
FailIfLanguageUnknown=TRUE
```

You must set the parameter `LanguageDetectionDirectory` to the path of the folder that contains the file `langdetect.dat`. The remaining parameters are optional. The parameter `OutputField` specifies the name of the document field to write the name of detected language to. By default, CFS rejects documents where it cannot detect a language but you can configure this by setting `FailIfLanguageUnknown`. To continue processing documents when a language cannot be detected, set `FailIfLanguageUnknown=FALSE`.

Translate Documents

CFS can use third-party translation services to translate documents into other languages. This can be useful when you have documents that are not in your users' native language.

> **IMPORTANT:**
> To perform translation, you must have a license for one of the supported translation services. CFS relies on third-party services to translate text between languages. The language translation library that CFS uses to communicate with third-party translation services is not included in the standard CFS installation but can be obtained through Micro Focus software support.

Micro Focus recommends that you configure the `LanguageTranslation` task as a post-import task. Use the `Post` parameter to specify the name of a section that contains the task settings.

The following is an example task that uses the translation services provided by an SDL Enterprise Translation Server:

```
[ImportTasks]
Post0=Library:LanguageTranslation

[LanguageTranslation]
Library=LanguageTranslation
TranslatorType=SdlEts
ApiBaseUrl=https://host:port
```

```
ApiKey=0a12b34c56d78e9
SourceField=DRECONTENT
TargetField=DRECONTENT
TargetLanguage=ENG
Quality=medium-high
```

The following is an example task that uses the translation services provided by sdlbeglobal.com:

```
[ImportTasks]
Post0=Library:LanguageTranslation

[LanguageTranslation]
Library=LanguageTranslation
TranslatorType=SdlBeGlobal
AccountId=12345
ApiKey=0a12b34c56d78e9
UserId=23456
TouchpointId=34567
SourceField=DRECONTENT
TargetField=DRECONTENT
TargetLanguage=ENG
```

In both cases, the `TranslatorType` configuration parameter specifies the type of translation service to use. To use an SDL Enterprise Translation Server, set this parameter to `SdlEts`. To use the services that are provided by `sdlbeglobal.com`, set this parameter to `SdlBeGlobal`. You must then set the relevant parameters that are required to use the API:

- To use an SDL Enterprise Translation Server, set the configuration parameter `ApiBaseUrl` to the base URL of the API to use for translation, and the `ApiKey` parameter to the API key.
- To use the services provided by sdlbeglobal.com, set the parameters `AccountId`, `ApiKey`, `UserId` and `TouchpointId` to the values provided by SDL.

The remaining parameters are optional but you can set these to customize the task to your use case.

The `SourceField` configuration parameter specifies the name of the document field to translate, and the `TargetField` parameter specifies the name of the field to contain the results. If you specify the same field name for both parameters, CFS reads the text from the field, sends it for translation, and then writes the result back to the same field (which overwrites the original value).

The source language is detected automatically, but if automatic detection is not successful you can specify the source language by setting the configuration parameter `SourceLanguage`, which accepts the name of a language or a three-letter language code. Alternatively, you can configure the task to read the source language from a document field, by setting the parameter `SourceLanguageField`. For information about the languages that are supported, refer to the SDL documentation or translation service user interface.

The target language is specified by the configuration parameter `TargetLanguage`, which has a default value of `ENG` (English).

If you are using an SDL Enterprise Translation Server, you can also choose how to balance translation quality and speed by setting the `Quality` parameter.

For more information about the configuration parameters that you can use to configure the language translation task, refer to the *Connector Framework Server Reference*.

Chapter 8: Index Documents

This section describes how to configure indexing.

Introduction

The final step in the ingestion process is to index information into an index, such as IDOL Server or Haven OnDemand. After CFS finishes processing documents, it automatically indexes them into the indexes that you have configured.

CFS can index documents into:

- **IDOL Server**.

  Index documents into IDOL Server to search, analyze, and find patterns in unstructured information. You can index documents directly into an IDOL Server, or send them to a Distributed Index Handler (DIH) to be distributed between multiple IDOL Servers in a distributed architecture.
- **Haven OnDemand**.

  Haven OnDemand analyzes unstructured information in the cloud.
- **Vertica**.

  Index documents into a Vertica database to analyze the structured information contained in your data repositories. Much of the metadata extracted by connectors and by KeyView is structured information held in structured fields, so you can use Vertica to gain insight into this information.

By default, CFS indexes each document into all of the indexes specified by the `IndexerSections` parameter in the `[Indexing]` section of its configuration file. However, if the document field `AUTN_INDEXER_SECTIONS` is set, CFS routes the document to the indexes specified in the field. The field accepts a comma-separated list of index names that must match the names of the sections in the CFS configuration file.

Configure the Batch Size and Time Interval

CFS indexes documents in batches. This is more efficient because fewer requests are made to the server.

Documents wait in the index queue until there are enough documents to create a batch, or until the maximum time interval for indexing is reached. If the time interval is reached, CFS indexes all of the documents in the queue regardless of the batch size.

To configure indexing settings

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, set the following configuration parameters:

`IndexBatchSize`	The number of documents to index in a single batch. CFS waits until this number of documents are ready for indexing, unless the `IndexTimeInterval` is reached.
`IndexTimeInterval`	The maximum amount of time, in seconds, that a document can wait in the index queue.

   For example:

   ```
   [Indexing]
   IndexBatchSize=1000
   IndexTimeInterval=600
   ```

4. Save the configuration file.

Index Documents into an IDOL Server

To index documents into an IDOL Server

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, use the `IndexerSections` parameter to specify the names of the sections that contain indexing settings. If this parameter is already set, add the name of the new indexer to the list. For example:

   ```
   [Indexing]
   IndexerSections=IdolServer
   ```

4. Create a new section in the CFS configuration file, with the same name that you specified in the `IndexerSections` parameter. In the new section, set the following parameters:

`Host`	The host name or IP address of the IDOL Server.

`Port`	The ACI Port of the IDOL Server.
`DefaultDatabaseName`	The name of the IDOL database to index a document into when the `DREDBNAME` document field is not set.
`SSLConfig`	(Optional) The name of a section in the CFS configuration file that contains SSL settings for connecting to IDOL. Set this parameter if your IDOL Server is configured to accept connections over SSL. For more information about the configuration parameters you can use to configure SSL connections, refer to the *Connector Framework Server Reference*.
`CreateDatabase`	(Optional, default `false`) Specifies whether IDOL should create databases that do not already exist. For example, if you set this parameter to `TRUE` and the database specified in a `DREDBNAME` document field does not exist, IDOL Server will create it.

For example:

```
[IdolServer]
Host=idol
Port=9000
DefaultDatabaseName=News
SSLConfig=SSLOptions

[SSLOptions]
SSLMethod=SSLV23
```

For more information about these parameters and other parameters that you can set to customize the indexing process, refer to the *Connector Framework Server Reference*.

5. Save and close the configuration file.

Index Documents into Haven OnDemand

CFS can index documents into a Haven OnDemand text index, or send the documents to a Haven OnDemand combination which can perform additional processing and then index the documents into a text index.

Prepare Haven OnDemand

Before you can send documents to Haven OnDemand, you must create a text index. For information about how to create text indexes, refer to the Haven OnDemand documentation.

Before you can send documents to a Haven OnDemand combination endpoint, the combination must exist. CFS requires your combination to accept the following input parameters, and produce the following output.

Input Parameters

Name	Type	Description
`json`	any	A JSON object that contains a single attribute 'documents' that is an array of document objects.
`index`	string	The name of the text index that you want the combination to add documents to. CFS uses the value of the parameter `HavenOnDemandIndexName` to set this value.
`duplicate_mode`	string	Specifies how to handle duplicates when adding documents to the text index. CFS uses the value of the parameter `HavenOnDemandDuplicateMode` to set this value.

Output

Name	Type	Description
`result`	any	The result of the call to `AddToTextIndex` made by the combination.

Configure CFS to Index into Haven OnDemand

This section describes how to send documents to Haven OnDemand.

To index documents into Haven OnDemand

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, use the `IndexerSections` parameter to specify the names of the sections that contain indexing settings. If this parameter is already set, add the name of the new indexer to the list. For example:

```
[Indexing]
IndexerSections=IdolServer,HavenOnDemand
```

4. Create a new section in the CFS configuration file, with the same name that you specified in the `IndexerSections` parameter. In the new section, set the following parameters:

`HavenOnDemandApiKey`	Your Haven OnDemand API key. You can obtain the key from your Haven OnDemand account.
`HavenOnDemandIndexName`	The name of the Haven OnDemand text index to index documents into.
`HavenOnDemandDuplicateMode`	The value to use for the `duplicate_mode` parameter in calls to the Haven OnDemand Add to Text Index API.
`SSLConfig`	The name of a section in the CFS configuration file that contains SSL settings for connecting to Haven OnDemand. The connection to Haven OnDemand must be made over

	SSL. For more information about the configuration parameters you can use to configure SSL connections, refer to the *Connector Framework Server Reference*.
`HavenOnDemandCombinationName`	(Optional) The name of the Haven OnDemand combination to send documents to. If you set this parameter, CFS sends all documents to the combination endpoint. If you don't set this parameter, CFS indexes all documents directly into the text index specified by `HavenOnDemandIndexName`.

For example:

```
[HavenOnDemand]
HavenOnDemandApiKey=[Your API Key]
HavenOnDemandIndexName=MyTextIndex
SSLConfig=SSLOptions
HavenOnDemandCombinationName=MyCombination

[SSLOptions]
SSLMethod=TLSV1
```

For more information about these parameters and other parameters that you can set to customize the indexing process, refer to the *Connector Framework Server Reference*.

5. Save and close the configuration file.

Index Documents into Vertica

CFS can index documents into Vertica, so that you can run queries on structured fields (document metadata).

Depending on the metadata contained in your documents, you could:

- Investigate the average age of documents in a repository. You might want to answer questions such as: How much time has passed since the documents were last updated? How many files are regularly updated? Does this represent a small proportion of the total number of documents? Who are the most active users?
- Find the number of e-mail messages sent to your sales or support teams each week, and calculate the average response time to customer queries.

Prerequisites

- CFS supports indexing into Vertica 7.1 and later.
- You must install the appropriate Vertica ODBC drivers (version 7.1 or later) on the machine that hosts Connector Framework Server. If you want to use an ODBC Data Source Name (DSN) in your connection string, you will also need to create the DSN. For more information about installing Vertica ODBC drivers and creating the DSN, refer to the Vertica documentation.

New, Updated and Deleted Documents

When documents are indexed into Vertica, CFS adds a timestamp that contains the time when the document was indexed. The field is named `VERTICA_INDEXER_TIMESTAMP` and the timestamp is in the format `YYYY-MM-DD HH:NN:SS`.

When a document in a data repository is modified, CFS adds a new record to the database with a new timestamp. All of the fields are populated with the latest data. The record describing the older version of the document is not deleted. You can create a projection to make sure your queries only return the latest record for a document.

When a connector detects that a document has been deleted from a repository, CFS inserts a new record into the database. The record contains only the `DREREFERENCE` and the field `VERTICA_INDEXER_DELETED` set to `TRUE`.

Fields, Sub-Fields, and Field Attributes

Documents that are created by connectors and processed by CFS can have multiple levels of fields, and field attributes. A database table has a flat structure, so this information is indexed into Vertica as follows:

- Document fields become columns in the flex table. An IDOL document field and the corresponding database column have the same name.
- Sub-fields become columns in the flex table. A document field named `my_field` with a sub-field named `subfield` results in two columns, `my_field` and `my_field.subfield`.
- Field attributes become columns in the flex table. A document field named `my_field`, with an attribute named `my_attribute` results in two columns, `my_field` holding the field value and `my_field.my_attribute` holding the attribute value.

Prepare the Vertica Database

Indexing documents into a standard database is problematic, because documents do not have a fixed schema. A document that represents an image has different metadata fields to a document that represents an e-mail message. Vertica databases solve this problem with *flex tables*. You can create a flex table without any column definitions, and you can insert a record regardless of whether a referenced column exists.

You must create a flex table before you index data into Vertica.

When creating the table, consider the following:

- Flex tables store entire records in a single column named `__raw__`. The default maximum size of the `__raw__` column is 128K. You might need to increase the maximum size if you are indexing documents with large amounts of content. Alternatively, you could configure CFS to remove content from documents before they are indexed.
- Documents are identified by their `DREREFERENCE`. Micro Focus recommends that you do not restrict the size of any column that holds this value, because this could result in values being truncated. As a result, rows that represent different documents might appear to represent the same document. If

you do restrict the size of the `DREREFERENCE` column, ensure that the length is sufficient to hold the longest `DREREFERENCE` that might be indexed.

To create a flex table without any column definitions, run the following query:

```
create flex table my_table();
```

To improve query performance, create real columns for the fields that you query frequently. For documents indexed by CFS, this is likely to include the `DREREFERENCE`:

```
create flex table my_table(DREREFERENCE varchar NOT NULL);
```

You can add new column definitions to a flex table at any time. Vertica automatically populates new columns with values for existing records. The values for existing records are extracted from the `__raw__` column.

For more information about creating and using flex tables, refer to the Vertica Documentation or contact Micro Focus Vertica technical support.

Configure CFS to Index into Vertica

The following procedure demonstrates a basic configuration that indexes all documents into a Vertica database.

However, you can customize the indexing process. For example, your CFS might be importing files from a File System Connector, e-mail messages from Exchange, and social media content. You might want to index these items into separate flex tables. To do this, you could run a Lua script to set the `AUTN_INDEXER_SECTIONS` field in each document, and create a separate indexing operation for each type of content.

To configure CFS to index documents into Vertica

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, use the `IndexerSections` parameter to specify the names of the sections that contain indexing settings. If this parameter is already set, add the name of the new indexer to the list. For example:

   ```
   [Indexing]
   IndexerSections=IdolServer,Vertica
   ```

4. Create a new section in the CFS configuration file, with the same name that you specified in the `IndexerSections` parameter. In the new section, set the following parameters:

`IndexerType`	The type of index to index documents into. Set this parameter to **`Library`**.
`LibraryDirectory`	The directory that contains the library to use to index data.
`LibraryName`	The name of the library to use to index data. You can omit the `.dll` or `.so` file extension. Set this parameter to **`verticaIndexer`**.
`ConnectionString`	The connection string to use to connect to the Vertica database.
`TableName`	The name of the table in the Vertica database to index the documents into.

The table must be a flex table and must exist before you start indexing documents. For more information, see Prepare the Vertica Database, on page 106.

For example:

```
[Vertica]
IndexerType=Library
LibraryDirectory=indexerdlls
LibraryName=verticaIndexer
ConnectionString=DSN=VERTICA
TableName=my_flex_table
```

For more information about these parameters and other parameters that you can set to customize the indexing process, refer to the *Connector Framework Server Reference*.

5. Save and close the configuration file.

Troubleshooting

This section describes how to troubleshoot problems that might occur when you index data into Vertica.

Documents are not indexed into Vertica

Documents cannot be indexed when the Vertica database server is unavailable, or cannot be reached by CFS. To see whether an indexing error has occurred, check the CFS indexer log. The default location for this log file is `logs/indexer.log`. If documents were not indexed successfully, you will need to ingest these documents again.

Index Documents into another CFS

You can index documents into another CFS. You might want to do this if you want to perform further processing on them.

To index documents into another CFS

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, use the `IndexerSections` parameter to specify the names of the sections in the configuration file that contain indexing settings. If this parameter is already set, add the name of the new indexer to the list. For example:

```
[Indexing]
IndexerSections=IndexCFS
```

4. Create a new section in the CFS configuration file, with the same name that you specified in the `IndexerSections` parameter. In the new section, set the following parameters:

`IndexerType`	The type of index that you want to index documents into. Set this parameter to **`CFS`**.
`Host`	The host name or IP address of the CFS.
`Port`	The ACI Port of the CFS.
`SSLConfig`	(Optional) The name of a section in the CFS configuration file that contains SSL settings for connecting to the other CFS. Set this parameter if the CFS is configured to accept connections over SSL. For more information about the configuration parameters you can use to configure SSL connections, refer to the *Connector Framework Server Reference*.

For example:

```
[IndexCFS]
IndexerType=CFS
Host=cfs.domain.com
Port=7000
SSLConfig=SSLOptions

[SSLOptions]
SSLMethod=TLSV1.2
```

For more information about these parameters and other parameters that you can set to customize the indexing process, refer to the *Connector Framework Server Reference*.

5. Save and close the configuration file.

Index Documents into MetaStore

To index documents into MetaStore

1. Stop CFS.
2. Open the CFS configuration file.
3. In the `[Indexing]` section, use the `IndexerSections` parameter to specify the names of the sections that contain indexing settings. If this parameter is already set, add the name of the new indexer to the list. For example:

```
[Indexing]
IndexerSections=IdolServer,MetaStore
```

4. Create a new section in the CFS configuration file, with the same name that you specified in the `IndexerSections` parameter. In the new section, set the following parameters:

`IndexerType`	Set this parameter to **`MetaStore`**.
`Host`	The host name or IP address of the MetaStore.
`Port`	The port of the MetaStore.

`SSLConfig`	(Optional) The name of the section in the CFS configuration file that contains the SSL settings to use for communicating with the MetaStore.

For example:

```
[MetaStore]
IndexerType=MetaStore
Host=localhost
Port=4500
```

For more information about these parameters and other parameters that you can set to customize the indexing process, refer to the *Connector Framework Server Reference*.

5. Save and close the configuration file.

Document Fields for Indexing

To customize the way that documents are indexed, set the following document fields.

AUTN_NO_INDEX

Documents that have this field are not indexed. You can use this field when you want to troubleshoot the ingestion process without indexing documents.

AUTN_INDEXER_SECTIONS

A comma-separated list of sections in the CFS configuration file to use to index the document. CFS indexes the document into all of the indexes that you specify. If this field is not set, CFS indexes the document into all of the indexes specified by the configuration parameter `IndexerSections`.

AUTN_INDEXPRIORITY

This field can be used to increase the priority of an index action sent to IDOL Server to index a batch of documents. You can specify a priority from 0 to 100, where 0 is the lowest priority and 100 is the highest. This means that you can configure some documents to be indexed before others, or before documents from sources other than CFS.

CAUTION:
Use this field with care. Modifying the index priority for documents changes the order of the index commands processed by IDOL. For example, in the case of an ingest-replace, the add command could be processed before the delete, resulting in a loss of data. Micro Focus recommends that you configure the indexing priority for batches of documents indexed into IDOL using the `IndexPriority` configuration parameter. This ensures that all of the batches indexed by CFS have the same priority.

If documents in a batch contain the field `INDEXPRIORITY`, and the value of this field is greater than that specified by the `IndexPriority` configuration parameter, the priority of the batch is increased to the highest `INDEXPRIORITY` field value present in the batch.

Manipulate Documents Before Indexing

CFS can index documents into multiple indexes. Normally, CFS indexes identical data into every index, but you might want to manipulate documents depending on the index that they are sent to. For example, if you are using Vertica to analyze structured information, you might want to remove the content from the documents indexed into Vertica, but keep the content in documents that are indexed into IDOL.

You cannot use import and index tasks to manipulate documents in this way, because those tasks affect documents sent to all of the indexes. To manipulate the documents sent to a single index, you can run a Lua script during the indexing process.

The script must define a `handler` function:

```
function handler(document, operation)
     -- do something, for example
     document:deleteField("UNINTERESTING_FIELD")
     return true
 end
```

The `operation` argument specifies the documents that you want to run the script on. This argument is a string and can be set to `add`, `update`, or `remove`:

- `add` - manipulate documents that are being added to the index. Ingest-adds are sent when a connector finds new documents in a repository, or when a document's content is changed (the old document is removed, and the new document added).
- `update` - manipulate documents that represent metadata updates.
- `remove` - manipulate documents that represent information deleted from the source repository.

To index the document the `handler` function must return `true`. To discard the document, return `false`.

To manipulate documents before indexing

1. Open the CFS configuration file.
2. In a section of the configuration file specified by the `IndexerSections` configuration parameter, set the `IndexLuaScript` parameter. This parameter specifies the path to the script that you want to run. For example:

   ```
   [Indexing]
   IndexerSections=IdolServer,Vertica

   [Vertica]
   IndexerType=Library
   LibraryDirectory=indexerdlls
   LibraryName=verticaIndexer
   ConnectionString=DSN=VERTICA
   ```

```
TableName=my_flex_table
IndexLuaScript=./scripts/remove_content.lua
```

3. Save and close the configuration file.

Chapter 9: Monitor Connector Framework Server

This section describes how to monitor CFS.

Use the Logs

As Connector Framework Server runs, it outputs messages to log files. Most log messages occur due to normal operation, for example when CFS starts, receives actions, or processes documents. If CFS encounters an error, the logs are the first place to look for information to help troubleshoot the problem.

CFS separates log messages into the following message types, each of which relates to a specific feature:

Log Message Type	Description
`Action`	Logs actions that are received by CFS, and related messages.
`Application`	Logs application-related occurrences, such as when the CFS starts.
`Import`	Information about the import process. When you set `LogLevel` to `FULL`, CFS can log a significant amount of information about the import process, which might reduce performance.
`Indexer`	Information about the indexing process.

Customize Logging

You can customize logging by setting up your own *log streams*. Each log stream creates a separate log file in which specific types of message are logged.

To set up log streams

1. Open the CFS configuration file in a text editor.
2. Find the `[Logging]` section. If the configuration file does not contain a `[Logging]` section, create it.
3. In the `[Logging]` section, create a list of the log streams you want to set up using the format *`N=LogStreamName`*. List the log streams in consecutive order, starting from 0 (zero). For example:

```
[Logging]
LogLevel=NORMAL
0=ApplicationLogStream
1=ActionLogStream
2=ImportLogStream
3=IndexLogStream
```

You can also use the `[Logging]` section to configure any default values for logging configuration parameters, such as `LogLevel`. For more information, refer to the *Connector Framework Server Reference*.

4. Create a new section for each of the log streams. Each section must have the same name as the log stream. For example:

   ```
   [ApplicationLogStream]
   [ActionLogStream]
   ...
   ```

5. Specify the settings for each log stream in the appropriate section. You can specify the type of logging to perform (for example, full logging), the maximum size of log files, and so on. For example:

   ```
   [ApplicationLogStream]
   LogTypeCSVs=application
   LogFile=application.log
   LogHistorySize=50
   LogTime=True
   LogEcho=False
   LogMaxSizeKBs=1024

   [ActionLogStream]
   LogTypeCSVs=action
   LogFile=action.log
   LogHistorySize=50
   LogTime=true
   LogEcho=false
   LogMaxSizeKBs=1024
   ```

6. Save and close the configuration file.
7. Restart CFS for your changes to take effect. For information about how to start and stop CFS, see Start and Stop Connector Framework Server, on page 29.

Monitor Asynchronous Actions using Event Handlers

Some of the actions that you can send to Connector Framework Server are asynchronous. Asynchronous actions do not run immediately, but are added to a queue. This means that the person or application that sends the action does not receive an immediate response. However, you can configure Connector Framework Server to call an event handler when an asynchronous action starts, finishes, or encounters an error.

You might use an event handler to:

- Return data about an event back to the application that sent the action.
- Write event data to a text file, to log any errors that occur.

You can also use event handlers to monitor the size of asynchronous action queues. If a queue becomes full this might indicate a problem, or that applications are making requests to Connector Framework Server faster than they can be processed.

Connector Framework Server can call an event handler for the following events.

OnStart	The `OnStart` event handler is called when Connector Framework Server starts processing an asynchronous action.
OnFinish	The `OnFinish` event handler is called when Connector Framework Server successfully finishes processing an asynchronous action.
OnError	The `OnError` event handler is called when an asynchronous action fails and cannot continue.
OnQueueEvent	The `OnQueueEvent` handler is called when an asynchronous action queue becomes full, becomes empty, or the queue size passes certain thresholds. • A `QueueFull` event occurs when the action queue becomes full. • A `QueueFilling` event occurs when the queue size exceeds a configurable threshold (`QueueFillingThreshold`) and the last event was a `QueueEmpty` or `QueueEmptying` event. • A `QueueEmptying` event occurs when the queue size falls below a configurable threshold (`QueueEmptyingThreshold`) and the last event was a `QueueFull` or `QueueFilling` event. • A `QueueEmpty` event occurs when the action queue becomes empty.

Connector Framework Server supports the following types of event handler:

- The `TextFileHandler` writes event data to a text file.
- The `HttpHandler` sends event data to a URL.
- The `LuaHandler` runs a Lua script. The event data is passed into the script.

Configure an Event Handler

To configure an event handler, follow these steps.

To configure an event handler

1. Stop Connector Framework Server.
2. Open the Connector Framework Server configuration file in a text editor.
3. Set the `OnStart`, `OnFinish`, `OnError`, or `OnQueueEvent` parameter to specify the name of a section in the configuration file that contains the event handler settings.
    - To run an event handler for all asynchronous actions, set these parameters in the `[Actions]` section. For example:

```
[Actions]
OnStart=NormalEvents
OnFinish=NormalEvents
OnError=ErrorEvents
```

- To run an event handler for a specific action, set these parameters in the `[`*`ActionName`*`]` section, where *`ActionName`* is the name of the action. The following example calls an event handler when the *`Example`* action starts and finishes successfully, and uses a different event handler to monitor the queue size:

  ```
  [Example]
  OnStart=NormalEvents
  OnFinish=NormalEvents
  OnQueueEvent=QueueSizeEvents
  ```

4. Create a new section in the configuration file to contain the settings for your event handler. You must name the section using the name you specified with the `OnStart`, `OnFinish`, `OnError`, or `OnQueueEvent` parameter.
5. In the new section, set the `LibraryName` parameter.

`LibraryName`	The type of event handler to use to handle the event. • To write event data to a text file, set this parameter to **`TextFileHandler`**, and then set the `FilePath` parameter to specify the path of the file. • To send event data to a URL, set this parameter to **`HttpHandler`**, and then use the HTTP event handler parameters to specify the URL, proxy server settings, credentials, and so on. • To run a Lua script, set this parameter to **`LuaHandler`**, and then use the `LuaScript` parameter to specify the script to run. For information about writing the script, see Write a Lua Script to Handle Events, on the next page.

For example:

```
[NormalEvents]
LibraryName=TextFileHandler
FilePath=./events.txt

[ErrorEvents]
LibraryName=HTTPHandler
URL=http://handlers:8080/lo-proxy/callback.htm?

[QueueSizeEvents]
LibraryName=LuaHandler
LuaScript=./handle_queue_events.lua
```

6. Save and close the configuration file. You must restart Connector Framework Server for your changes to take effect.

Write a Lua Script to Handle Events

The Lua event handler runs a Lua script to handle events. The Lua script must contain a function named `handler` with the arguments `request` and `xml`, as shown below:

```
function handler(request, xml)
       ...
end
```

- `request` is a table holding the request parameters. For example, if the request was `action=Example&MyParam=Value`, the table will contain a key `MyParam` with the value `Value`. Some events, for example queue size events, are not related to a specific action and so the table might be empty.
- `xml` is a string of XML that contains information about the event.

Monitor the size of the Import and Index Queues

CFS generates events when the import queue and the outgoing (indexing) queue become full, become empty, or the queue size passes certain thresholds. If a queue approaches its maximum size, this might indicate a problem, or that applications are making requests to Connector Framework Server faster than they can be processed.

CFS generates the following events for each queue that is monitored:

- A `QueueFull` event occurs when the queue becomes full.
- A `QueueFilling` event occurs when the queue size exceeds a configurable threshold (`QueueFillingThreshold`) and the last event was a `QueueEmpty` or `QueueEmptying` event.
- A `QueueEmptying` event occurs when the queue size falls below a configurable threshold (`QueueEmptyingThreshold`) and the last event was a `QueueFull` or `QueueFilling` event.
- A `QueueEmpty` event occurs when the queue becomes empty.

You can configure event handlers to process these events. For example, you might want to notify an administrator if the size of a queue reaches 80 percent of the maximum.

To monitor queue sizes

1. Stop CFS.
2. Open the CFS configuration file in a text editor.
3. Set the `OnQueueEvent` parameter to the name of a section that configures the event handler.
    - To monitor the size of the import queue, set this parameter in the `[ImportService]` section. For example:

      ```
      [ImportService]
      OnQueueEvent=MyEventHandler
      ```

    - To monitor the size of the outgoing (indexing) queue, set this parameter in the `[Indexing]` section. For example:

```
[Indexing]
OnQueueEvent=MyEventHandler
```

4. Create a new section in the configuration file to contain the settings for your event handler. You must name the section using the name you specified with the `OnQueueEvent` parameter.
5. In the new section, set the `LibraryName` parameter.

`LibraryName`	The type of event handler to use to handle the event. • To write event data to a text file, set this parameter to **`TextFileHandler`**, and then set the `FilePath` parameter to specify the path of the file. • To send event data to a URL, set this parameter to **`HttpHandler`**, and then use the HTTP event handler parameters to specify the URL, proxy server settings, credentials, and so on. • To run a Lua script, set this parameter to **`LuaHandler`**, and then use the `LuaScript` parameter to specify the script to run. For information about writing the script, see Write a Lua Script to Handle Events, on the previous page.

   For example:

```
[MyEventHandler]
LibraryName=LuaHandler
LuaScript=./handle_queue_event.lua
```

6. Save and close the configuration file. You must restart CFS for your changes to take effect.

Set Up Document Tracking

Document tracking reports metadata about documents when they pass through various stages in the ingestion and indexing process. Document tracking can help you detect problems with the indexing process.

You can write document tracking events to a database, log file, or IDOL Server. For information about how to set up a database to store document tracking events, refer to the *IDOL Server Administration Guide*.

To enable Document Tracking

1. Open the CFS configuration file.
2. Create a new section in the configuration file, named `[DocumentTracking]`.
3. In the `[DocumentTracking]` section, specify where the document tracking events are sent.

- To send document tracking events to a database through ODBC, set the following parameters:

`Backend`	To send document tracking events to a database, set this parameter to **`Library`**.
`LibraryPath`	Specify the location of the ODBC document tracking library. This is included with IDOL Server.
`ConnectionString`	The ODBC connection string for the database.

  For example:

  ```
  [DocumentTracking]
  Backend=Library
  LibraryPath=C:\Autonomy\IDOLServer\IDOL\modules\dt_odbc.dll
  ConnectionString=DSN=MyDatabase
  ```

- To send document tracking events to the CFS import log, set the following parameters:

`Backend`	To send document tracking events to the logs, set this parameter to **`Log`**.
`DatabaseName`	The name of the log stream to send the document tracking events to. Set this parameter to **`import`**.

  For example:

  ```
  [DocumentTracking]
  Backend=Log
  DatabaseName=import
  ```

- To send document tracking events to an IDOL Server, set the following parameters:

`Backend`	To send document tracking events to an IDOL Server, set this parameter to **`IDOL`**.
`TargetHost`	The host name or IP address of the IDOL Server.
`TargetPort`	The index port of the IDOL Server.

  For example:

  ```
  [DocumentTracking]
  Backend=IDOL
  TargetHost=idol
  TargetPort=9001
  ```

For more information about the parameters you can use to configure document tracking, refer to the *Connector Framework Server Reference*.

4. Save and close the configuration file.

Appendix A: KeyView Supported Formats

This section lists information about the file formats that can be detected and processed by KeyView.

Supported Formats

The tables in this section provide the following information:

- The file formats that can be processed by KeyView, and the features that are supported for each format:
  - The **Filter** column specifies whether KeyView can extract the main content of the file (for example the text in a document or the body of an email message). CFS writes this text to the document content.
  - The **Extract** column specifies whether KeyView can extract subfiles from the file, if it is a container file.
- The file formats for which KeyView can detect and extract the character set and metadata information (properties such as title, author, and subject).

  Even though a file format might be able to provide character set information, some documents might not contain character set information. Therefore, the document reader would not be able to determine the character set of the document. In this case, either the operating system code page or the character set specified in the API is used.
- The document reader used to filter each format.

Key to Support Tables

Symbol	Description
Y	The format is supported. You can extract metadata for this format. You can determine the character set for this format.
N	The format is not supported. You cannot extract metadata for this format. You cannot determine the character set for this format.
P	Partial metadata is extracted from this format. Some non-standard fields are not extracted.
T	Only text is extracted from this format. Formatting information is not extracted.
M	Only metadata (title, subject, author, and so on) is extracted from this format. Text and formatting information are not extracted.

Archive Formats

Supported Archive Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
7-Zip	4.57	z7zsr, multiarcsr[1]	7Z	N	N	Y	Y	N	n/a	N
AD1	n/a	ad1sr	AD1	N	N	Y	Y	N	n/a	N
ARJ	n/a	multiarcsr	ARJ	N	N	N	Y	N	n/a	N
B1	n/a	b1sr	B1	N	N	Y	Y	N	n/a	N
BinHex	n/a	kvhqxsr	HQX	N	N	Y	Y	N	n/a	N
Bzip2	n/a	bzip2sr	BZ2	N	N	Y	Y	N	n/a	N
Expert Witness Compression Format (EnCase)	6	encasesr	E01, L01	N	N	Y	Y	N	n/a	N
	7	encase2sr	Lx01	N	N	Y	Y	N	n/a	N
GZIP	2	kvgzsr	GZ	N	N	N	Y	N	n/a	N
		kvgz	GZ	N	N	Y	N	N	n/a	N
ISO	n/a	isosr	ISO	N	N	Y	Y	N	n/a	N
Java Archive	n/a	unzip	JAR	N	N	Y	Y	N	n/a	N
Legato EMailXtender	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N

[1] 7zip is supported with the multiarcsr reader on some platforms for Extract.

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Archive										
MacBinary	n/a	macbinsr	BIN	N	N	Y	Y	N	n/a	N
Mac Disk Copy Disk Image	n/a	dmgsr	DMG	N	N	Y	Y	N	n/a	N
Microsoft Backup File	n/a	bkfsr	BKF	N	N	Y	Y	N	n/a	N
Microsoft Cabinet format	1.3	cabsr	CAB	N	N	Y	Y	N	n/a	N
Microsoft Compiled HTML Help	3	chmsr	CHM	N	N	Y	Y	N	n/a	N
Microsoft Compressed Folder	n/a	lzhsr	LZH LHA	N	N	N	Y	N	n/a	N
PKZIP	through 9.0	unzip	ZIP	N	N	Y	Y	N	n/a	N
RAR archive	2.0 through 3.5	rarsr	RAR	N	N	N	Y	N	n/a	N
RAR5 archive	5	multiarcsr	RAR5	N	N	N	Y	N	n/a	N
Tape Archive	n/a	tarsr	TAR	N	N	Y	Y	N	n/a	N
UNIX Compress	n/a	kvzeesr	Z	N	N	N	Y	N	n/a	N
		kvzee	Z	N	N	Y	N	N	n/a	N
UUEncoding	all versions	uudsr	UUE	N	N	Y	Y	N	n/a	N
XZ	n/a	multiarcsr	XZ	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Windows Scrap File	n/a	olesr	SHS	N	N	N	Y	N	n/a	N
WinZip	through 10	unzip	ZIP	N	N	Y	Y	N	n/a	N

Binary Format

Supported Binary Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Executable	n/a	exesr	EXE	N	N	Y	N	N	n/a	N
Link Library	n/a	exesr	DLL	N	N	Y	N	N	n/a	N

Computer-Aided Design Formats

Supported CAD Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing	R13, R14, R15/2000, 2004,	kpODArdr kpDWGrdr[1]	DWG	Y	Y[2]	Y [3]	N	Y	Y	N

[1]On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.
[2]On non-Windows platforms, graphic rendering is supported through the kpDWGrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.
[3]On non-Windows platforms, graphic rendering is supported through the kpDWGrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
	2007, 2010, 2013									
AutoCAD Drawing Exchange	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDXFrdr[1]	DXF	Y	Y[2]	Y[3]	N	Y	Y	N
CATIA formats	5	kpCATrdr	CAT[4]	Y	N	N	N	Y	N	N
Microsoft Visio	4, 5, 2000, 2002, 2003, 2007, 2010[5]	vsdsr	VSD	Y	Y	Y	Y[6]	Y	Y	N
		kpVSD2rdr	VSD, VSS VST	Y	Y	Y	N	Y	Y	N
	2013	ActiveX components	VSDM VSSM VSTM VSDX VSSX VSTX	N	N	Y[7]	N	Y	N	N

[1]On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.
[2]On non-Windows platforms, graphic rendering is supported through the kpDXFrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.
[3]On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.
[4]All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.
[5]Viewing and Export use the graphic reader, kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions. Image fidelity in Viewing and Export is therefore only supported for versions 2003 and above. Filter uses the graphic reader kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions.
[6]Extraction of embedded OLE objects is supported for Filter on Windows platforms only.
[7]Visio 2013 is supported in Viewing only, with the support of ActiveX components from the Microsoft Visio 2013 Viewer. Image fidelity is supported but other features, such as highlighting, are not.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
		kpVSDXrdr	VSDM VSSM VSTM VSDX VSSX VSTX	Y	Y	Y[4]	Y	Y	Y	N
Unigraphics (UG) NX		kpUGrdr	PRT	Y	N	N	N	N	N	N

Database Formats

Supported Database Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
dBase Database	III+, IV	dbfsr	DBF	Y	Y	Y	N	N	N	N
Microsoft Access	95, 97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	mdbsr	MDB, ACCDB	Y	T	T	N	N	Y[1]	N
Microsoft Project	2000, 2002, 2003, 2007, 2010, 2013	mppsr	MPP	Y	Y	Y	Y	Y	Y	N

[1]Charset is not supported for Microsoft Access 95 or 97.

Desktop Publishing

Supported Desktop Publishing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Publisher	98 to 2016	mspubsr	PUB	Y	T	T	Y	Y	Y	N

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7	pdfsr	PDF	Y	Y	N	Y[1]	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N
		kppdfrdr	PDF	N	Y	Y	N	N	N	N
		kppdf2rdr[2]	PDF	N	N	Y	N	N	N	N

[1]Includes support for extraction of subfiles from PDF Portfolio documents.

[2]kppdf2rdr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.

Graphic Formats

Supported Graphic Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Computer Graphics Metafile	n/a	kpcgmrdr[1]	CGM	Y	Y	Y	N	N	N	N
CorelDRAW[2]	through 9.0 10, 11, 12, X3	kpcdrrdr	CDR	N	Y	Y	N	N	N	N
DCX Fax System	n/a	kpdcxrdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in Medicine (DICOM)	n/a	dcmsr	DCM	M	N	N	N	Y	N	N
Encapsulated PostScript (raster)	TIFF header	kpepsrdr	EPS	N	Y	Y	N	N	N	N
Enhanced Metafile	n/a	kpemfrdr	EMF	Y	Y	Y	N	Y	N	N
GIF	87, 89	kpgifrdr	GIF	N	Y	Y	N	N	N	N
		gifsr		M	M	N	N	Y	N	N
JBIG2	n/a	kpJBIG2rdr	JBIG2	N	Y	Y	N	N	N	N

[1]Files with non-partitioned data are supported.

[2]CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
JPEG	n/a	kpjpgrdr	JPEG	N	Y	Y	N	N	N	N
		jpgsr		M	M	N	N	Y	N	N
JPEG 2000	n/a	kpjp2000rdr	JP2, JPF, J2K, JPWL, JPX, PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
Lotus AMIDraw Graphics	n/a	kpsdwrdr	SDW	N	Y	Y	N	N	N	N
Lotus Pic	n/a	kppicrdr	PIC	Y	Y	Y	N	N	N	N
Macintosh Raster	2	kppctrdr	PIC PCT	N	Y	Y	N	N	N	N
MacPaint	n/a	kpmacrdr	PNTG	N	Y	Y	N	N	N	N
Microsoft Office Drawing	n/a	kpmsordr	MSO	N	Y	Y	N	N	N	N
Omni Graffle	n/a	kpGFLrdr	GRAFFLE	Y	N	N	N	Y	Y	N
PC PaintBrush	3	kppcxrdr	PCX	N	Y	Y	N	N	N	N
Portable Network Graphics	n/a	kppngrdr	PNG	N	Y	Y	N	N	N	N
		pngsr	PNG	M	M	N	N	Y	N	N
SGI RGB Image	n/a	kpsgirdr	RGB	N	Y	Y	N	N	N	N
Sun Raster Image	n/a	kpsunrdr	RS	N	Y	Y	N	N	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Tagged Image File	through 6.0[1]	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifrdr	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kpTGArdr	TGA	N	Y	Y	N	N	N	N
Windows Animated Cursor	n/a	kpanirdr	ANI	N	Y	Y	N	N	N	N
Windows Bitmap	n/a	kpbmprdr	BMP	N	Y	Y	N	N	N	N
		bmpsr	BMP	M	M	N	N	Y	N	N
Windows Icon Cursor	n/a	kpicordr	ICO	N	Y	Y	N	N	N	N
Windows Metafile	3	kpwmfrdr	WMF	Y	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N
WordPerfect Graphics 2	2, 7	kpwg2rdr	WPG	N	Y	Y	N	N	N	N

[1]The following compression types are supported: no compression, CCITT Group 3 1-Dimensional Modified Huffman, CCITT Group 3 T4 1-Dimensional, CCITT Group 4 T6, LZW, JPEG (only Gray, RGB and CMYK color space are supported), and PackBits.

Mail Formats

Supported Mail Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Documentum EMCMF	n/a	msgsr	EMCMF	N	N	Y	Y	Y	Y	N
Domino XML Language[1]	n/a	dxlsr	DXL	N	N	Y	Y	Y	N	N
GroupWise FileSurf	n/a	gwfssr	GWFS	N	N	Y	Y	Y	N	N
Legato Extender	n/a	onmsr	ONM	N	N	Y	Y	Y	N	N
Lotus Notes database	4, 5, 6.0, 6.5, 7.0, 8.0	nsfsr	NSF	N	N	Y	Y	Y	N	N
Mailbox[2]	Thunderbird 1.0, Eudora 6.2	mbxsr[3]	MBX	N	N	T	Y	Y	Y	N
Microsoft Entourage Database	2004	entsr	various	N	N	Y	Y	Y	Y	N

[1]Supports non-encrypted embedded files only.

[2]KeyView supports MBX files created by Eudora Email and Mozilla Thunderbird. MBX files created by other common mail applications are typically filtered, converted, and displayed.

[3]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	msgsr[1]	MSG, OFT	Y	T	T	Y	Y	Y [2]	N
Microsoft Outlook DBX	5.0, 6.0	dbxsr	DBX	N	N	Y	Y	Y	Y	N
Microsoft Outlook Express	Windows 6 MacIntosh 5	emlsr[3]	EML	Y	T	T	Y	Y	Y	N
		mbxsr[4]	EML	N	N	T	Y	Y	Y	N
Microsoft Outlook iCalendar	1.0, 2.0	icssr	ICS, VCS	N	N	Y	Y	Y	Y	N
Microsoft Outlook for Macintosh	2011	olmsr	OLM	N	N	Y	Y	N	Y	N
Microsoft Outlook Offline Storage File	97, 2000, 2002, 2003, 2007, 2010, 2013	pffsr[5]	OST	N	N	Y	Y	Y	Y	N

[1]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[2]Returns "Unicode" character set for version 2003 and up, and "Unknown" character set for previous versions.

[3]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[4]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[5]The reader `pffsr` is available only on Windows and Linux.

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Outlook Personal Folder	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	pstsr[1][2]	PST	N	N	Y	Y	Y	N	N
	97, 2000, 2002, 2003, 2007, 2010, 2013	pstnsr	PST	N	N	Y	Y	Y	Y	N
Microsoft Outlook vCard Contact	2.1, 3.0, 4.0	vcfsr	VCF	Y	Y	T	N	Y	N	N
Text Mail (MIME)	n/a	emlsr[3]	various	Y	T	T	Y	Y	Y	N
		mbxsr[4]	various	Y	T	T	Y	Y	Y	N
Transport Neutral Encapsulation Format	n/a	tnefsr	various	N	N	Y	Y	Y	Y	N

Multimedia Formats

Viewing SDK plays some multimedia files using the Windows Media Control Interface (MCI). MCI is a set of Windows APIs that communicate with multimedia devices.

[1]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[2]Uses Microsoft Messaging Application Programming Interface (MAPI). Note that the native PST reader (`pstsr`) works only on Windows, and requires that you have Microsoft Outlook installed. As an alternative, the MAPI reader (`pstnsr`) runs on all platforms, and does not require Microsoft Outlook.

[3]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[4]This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

Supported Multimedia Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Advanced Systems Format	1.2	asfsr	ASF WMA WMV	N	N	N	N	Y	N	N
Audio Interchange File Format	n/a	MCI	AIFF	N	N	Y	N	N	N	N
		aiffsr	AIFF	M	N	N	N	Y	N	N
Microsoft Wave Sound	n/a	MCI	WAV	N	N	Y	N	N	N	N
		riffsr	WAV	M	N	N	N	Y	N	N
MIDI	n/a	MCI	MID	N	N	Y	N	N	N	N
MPEG-1 Audio layer 3	ID3 v1 and v2	MCI	MP3	N	N	Y	N	N	N	N
		mp3sr	MP3	M	M	Y	N	Y	N	N
MPEG-1 Video	2, 3	MCI	MPG	N	N	Y	N	N	N	N
MPEG-2 Audio	n/a	MCI	MPEGA	N	N	Y	N	N	N	N
MPEG-4 Audio	n/a	mpeg4sr	MP4 3GP	M	N	N	N	Y	N	N
NeXT/Sun Audio	n/a	MCI	AU	N	N	Y	N	N	N	N
QuickTime Movie	2, 3, 4	MCI	QT MOV	N	N	Y	N	N	N	N
Windows Video	2.1	MCI	AVI	N	N	Y	N	N	N	N

NOTE:
Depending on the default multimedia player installed on your computer, the View API might not be able to play some supported multimedia formats. To play multimedia files, the View API uses the Windows Media Control Interface (MCI) to communicate with the multimedia

player installed on your computer. If the player does not play a multimedia file that is supported by the Viewing SDK, the View API cannot play the file.

If you cannot play a supported multimedia file by using the View API, install a different multimedia player or compressor/decompressor (codec) component.

Presentation Formats

Supported Presentation Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Keynote	2, 3, ‘08, ‘09	kpIWPGrdr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16	kpIWPG13rdr	KEY	Y	N	N	N	N	N	N
Applix Presents	4.0, 4.2, 4.3, 4.4	kpagrdr	AG	Y	Y	Y	N	N	N	N
Corel Presentations	6, 7, 8, 9, 10, 11, 12, X3	kpshwrdr	SHW	Y	Y	Y	N	N	N	N
Extensible Forms Description Language	n/a	kpXFDLrdr	XFD XFDL	Y	Y	Y	N	Y	Y	N
Lotus Freelance Graphics	96, 97, 98, R9, 9.8	kpprzrdr	PRZ	Y	Y	Y	N	N	N	N
Lotus Freelance Graphics 2	2	kpprerdr	PRE	Y	Y	Y	N	N	N	N
Macromedia Flash	through 8.0	swfsr	SWF	Y	Y	Y	N	N	Y[1]	N

[1] The character set cannot be determined for versions 5.x and lower.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft OneNote	2007, 2010, 2013, 2016	kpONErdr	ONE ONETOC2	Y	Y	Y	Y	N	Y	N
Microsoft PowerPoint Macintosh	98	kpp40rdr	PPT	Y	Y	Y	N	N	N	N
	2001, v.X, 2004	kpp97rdr	PPT PPS POT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint PC	4	kpp40rdr	PPT	Y	Y	Y	N	P	N	N
Microsoft PowerPoint Windows	95	kpp95rdr	PPT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint Windows	97, 2000, 2002, 2003	kpp97rdr	PPT PPS POT	Y	Y	Y	Y	P	Y	Y[1]
Microsoft PowerPoint Windows XML	2007, 2010, 2013, 2016	kpppxrdr	PPTX PPTM POTX POTM PPSX PPSM PPAM	Y	Y	Y	Y	Y	Y	Y
OASIS Open	1, 2[2]	kpodfrdr	SXD	Y	Y	Y	Y[3]	Y	Y	N

[1]Slide footers are supported for Microsoft PowerPoint 97 and 2003.

[2]Generated by OpenOffice Impress 2.0, StarOffice 8 Impress, and IBM Lotus Symphony Presentation 3.0.

[3]Supported using the `olesr` embedded objects reader.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Document Format			SXI ODG ODP							
OpenOffice Impress, LibreOffice Impress	1 to 5	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N
StarOffice Impress	6, 7, 8, 9	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N

Spreadsheet Formats

Supported Spreadsheet Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Numbers	‘08, ‘09	iwsssr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16	iwss13sr	NUMBERS	Y	T	T	N	N	Y	N
Applix Spreadsheets	4.2, 4.3, 4.4	assr	AS	Y	Y	Y	N	N	Y	N
Comma Separated Values	n/a	csvsr	CSV	Y	Y	Y	N	N	N	N
Corel Quattro Pro	5, 6, 7, 8	qpssr	WB2 WB3	Y	Y	Y	N	P	Y	N
	X4	qpwsr	QPW	Y	N	Y	N	P	Y	N
Data Interchange	n/a	difsr		Y	Y	Y	N	N	N	N

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Format										
Lotus 1-2-3	96, 97, R9, 9.8	l123sr	123	Y	Y	Y	N	P	Y	N
Lotus 1-2-3	2, 3, 4, 5	wkssr	WK4	Y	Y	Y	N	N	Y	N
Lotus 1-2-3 Charts	2, 3, 4, 5	kpchtrdr	123	N	Y	Y	N	N	N	N
Microsoft Excel Charts	2, 3, 4, 5, 6, 7	kpchtrdr	XLS	N	Y	Y	N	N	N	N
Microsoft Excel Macintosh	98, 2001, v.X, 2004	xlssr	XLS	Y	Y	Y	Y[1]	Y	Y	N
Microsoft Excel Windows	2.2 through 2003	xlssr	XLS XLW XLT XLA	Y	Y	Y	Y[2]	Y	Y	Y
Microsoft Excel Windows XML	2007, 2010, 2013, 2016	xlsxsr	XLSX XLTX XLSM XLTM XLAM	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel Binary Format	2007, 2010, 2013, 2016	xlsbsr	XLSB	Y	Y	Y	N	N	N	N
Microsoft Works Spreadsheet	2, 3, 4	mwssr	S30 S40	Y	Y	Y	N	N	Y	N

[1]Supported using the embedded objects reader `olesr`.

[2]Supported for versions 97 and higher using the embedded objects reader `olesr`.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
OASIS Open Document Format	1, 2[1]	odfsssr	ODS SXC STC	Y	Y	Y	Y[2]	Y	Y	N
OpenOffice Calc, LibreOffice Calc	1 to 5	sosr	SXC ODS OTS	Y	T	T	N	Y	Y	N
StarOffice Calc	6, 7, 8, 9	sosr	SXC ODS	Y	T	T	N	Y	Y	N

Text and Markup Formats

Supported Text and Markup Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
ANSI	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
ASCII	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
HTML	3, 4	htmsr	HTM	Y	Y	Y	N	P	Y	N
Microsoft Excel Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Word Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N

[1]Generated by OpenOffice Calc 2.0, StarOffice 8 Calc, and IBM Lotus Symphony Spreadsheet 3.0.
[2]Supported using the embedded objects reader `olesr`.

Supported Text and Markup Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Visio XML	2003	xmlsr	VDX VTX	Y	T	T	N	Y	Y	N
MIME HTML	n/a	mhtsr	MHT	Y	Y	Y	N	Y	Y	N
Rich Text Format	1 through 1.7	rtfsr	RTF	Y	Y	Y	N	P	Y	Y
Unicode HTML	n/a	unihtmsr	HTM	Y	Y	Y	N	Y	Y	N
Unicode Text	3, 4	unisr	TXT	Y	Y	Y	N	N	Y	N
XHTML	1.0	htmsr	HTM	Y	Y	Y	N	Y	Y	N
XML (generic)	1.0	xmlsr	XML	Y	T	T	N	Y	Y	N

Word Processing Formats

Supported Word Processing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe FrameMaker Interchange Format	5, 5.5, 6, 7	mifsr	MIF	Y	Y	Y	N	N	Y	N
Apple iChat Log	1, AV 2 AV 2.1, AV 3	ichatsr	ICHAT	Y	Y	Y	N	N	N	N
Apple iWork Pages	‘08, ‘09	iwwpsr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16	iwwp13sr	PAGES	Y	T	T	N	N	N	N
Applix Words	3.11, 4, 4.1, 4.2, 4.3, 4.4	awsr	AW	Y	Y	Y	N	N	Y	Y

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Corel WordPerfect Linux	6.0, 8.1	wp6sr	WPS	Y	Y	Y	N	P	Y	N
Corel WordPerfect Macintosh	1.02, 2, 2.1, 2.2, 3, 3.1	wpmsr	WPM	Y	Y	Y	N	N	Y	N
Corel WordPerfect Windows	5, 5.1	wosr	WO	Y	Y	Y	N	P	Y	Y
Corel WordPerfect Windows	6, 7, 8, 9, 10, 11, 12, X3	wp6sr	WPD	Y	Y	Y	N	P	Y	Y
DisplayWrite	4	dw4sr	IP	Y	Y	Y	N	N	Y	N
Folio Flat File	3.1	foliosr	FFF	Y	Y	Y	N	Y	Y	Y
Founder Chinese E-paper Basic	3.2.1	cebsr[1]	CEB	Y	N	N	N	N	N	N
Fujitsu Oasys	7	oa2sr	OA2	Y	Y	Y	N	P	N	N
Haansoft Hangul	97	hwpsr	HWP	Y	N	N	N	N	Y	N
	2002, 2005, 2007, 2010	hwposr	HWP	Y	T	T	Y	Y	Y	N
Health level7	2.0	hl7sr	HL7	Y	Y	Y	N	Y	Y	N
IBM DCA/RFT (Revisable Form Text)	SC23-0758-1	dcasr	DC	Y	Y	Y	N	N	Y	N

[1]This reader is only supported on Windows 32-bit platforms.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
JustSystems Ichitaro	8 through 2013	jtdsr	JTD	Y	Y	Y	N	P	N	Y
Lotus AMI Pro	2, 3	lasr	SAM	Y	Y	Y	N	P	Y	Y
Lotus AMI Professional Write Plus	2.1	lasr	AMI	Y	Y	Y	N	N	N	Y
Lotus Word Pro	96, 97, R9	lwpsr	LWP	Y	Y	Y	N	P	N	Y
Lotus SmartMaster	96, 97	lwpsr	MWP	Y	Y	Y	N	N	N	N
Microsoft Word Macintosh	4, 5, 6, 98	mbsr	DOC	Y	Y	Y	N	Y	N	Y
	2001, v.X, 2004	mw8sr	DOC DOT	Y	Y	Y	Y[1]	Y	Y	N
Microsoft Word PC	4, 5, 5.5, 6	mwsr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	1.0 and 2.0	misr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	6, 7, 8, 95	mw6sr	DOC	Y	Y	Y	N	Y	Y	Y
Microsoft Word Windows	97, 2000, 2002, 2003	mw8sr	DOC DOT	Y	Y	Y	Y[2]	Y	Y	Y
Microsoft Word Windows XML	2007, 2010, 2013, 2016	mwxsr	DOCM DOCX	Y	Y	Y	Y	Y	Y	Y

[1] Supported using the embedded objects reader `olesr`.

[2] Supported using the embedded objects reader `olesr`.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
			DOTX DOTM							
Microsoft Word Windows Flat XML	2007, 2010, 2013, 2016	mwxsr	XML	Y	Y	Y	Y	Y	Y	Y
Microsoft Works	1, 2, 3, 4	mswsr	WPS	Y	Y	Y	N	N	N	Y
Microsoft Works	6, 2000	msw6sr	WPS	Y	Y	Y	N	N	N	Y
Microsoft Windows Write	1, 2, 3	mwsr	WRI	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2[1]	odfwpsr	ODT SXW STW	Y	Y	Y	Y[2]	Y	Y	Y
Omni Outliner	v3, OPML, OOutline	oo3sr	OO3 OPML OOUTLINE	Y	Y	Y	N	N	Y	N
OpenOffice Writer, LibreOffice Writer	1 to 5	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Open Publication Structure eBook	2.0, 3.0	epubsr	EPUB	Y	Y	Y	N	Y	Y	N
StarOffice Writer	6, 7, 8, 9	sosr	SXW ODT	Y	T	T	N	Y	Y	N

[1]Generated by OpenOffice Writer 2.0, StarOffice 8 Writer, and IBM Lotus Symphony Documents 3.0.

[2]Supported using the embedded objects reader `olesr`.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Skype Log	3	skypesr	DBB	Y	Y	Y	N	N	N	N
WordPad	through 2003	rtfsr	RTF	Y	Y	Y	N	P	Y	N
XML Paper Specification	n/a	xpssr	XPS	Y	T	T	N	N	N	N
XyWrite	4.12	xywsr	XY4	Y	Y	Y	N	N	N	N
Yahoo! Instant Messenger	n/a	yimsr[1]	DAT	Y	Y	Y	N	N	N	N

[1]To successfully use this reader, you must set the `KV_YAHOO_ID` environment variable to the Yahoo user ID. You can optionally set the `KV_OTHER_YAHOO_ID` environment variable to the other Yahoo user ID. If you do not set it, "`Other`" is used by default. If you enter incorrect values for the environment variables, erroneous data is generated.

Supported Formats (Detected)

The file formats listed in this section can be detected by the KeyView format detection module, but cannot be filtered, converted, or displayed.

These file formats therefore cannot be processed by CFS.

- 3D Systems STL format
- Ability Office (SS, DB, GR, WP, COM)
- AC3 audio
- ACT
- Adobe FrameMaker
- Adobe FrameMaker Markup Language
- AES Multiplus Comm
- Aldus Freehand (Macintosh)
- Aldus PageMaker (DOS)
- Aldus PageMaker (Macintosh)
- Amiga IFF-8SVX sound
- Amiga MOD sound
- Apple Binary Property List
- Apple Double
- Apple iWork
- Apple Photoshop Document
- Apple Single
- Apple XML Property List
- Appleworks
- Applix Alis
- Applix Asterix
- Applix Graphics
- ARC/PAK Archive
- ASCII-armored PGP encoded
- ASCII-armored PGP Public Keyring
- ASCII-armored PGP signed
- AutoDesk Animator FLIC Animation
- AutoDesk Animator Pro FLIC Animation
- AutoDesk WHIP
- AutoShade Rendering
- B1 Archive
- BlackBerry Activation File

- CADAM Drawing
- CADAM Drawing Overlay
- CCITT Group 3 1-Dimensional (G31D)
- COMET TOP Word
- Confifer Software WavPack
- Convergent Tech DEF Comm.
- Corel Draw CMX
- cpio Archive (UNIX/VAX/SUN)
- CPT Communication
- Creative Voice (VOC) sound
- Curses Screen Image (UNIX/VAX/SUN)
- Data Point VISTAWORD
- DCX Fax
- DEC WPS PLUS
- DECdx
- Desktop Color Separation (DCS)
- Device Independent file (DVI)
- DG CEOwrite
- DG Common Data Stream (CDS)
- DIF Spreadsheet
- Digital Document Interchange Format (DDIF)
- Digital Imaging and Communications in Medicine (DICOM)
- Disk Doubler Compression
- EBCDIC Text
- eFax
- ENABLE
- ENABLE Spreadsheet (SSF)
- Envoy (EVY)
- Executable UNIX/VAX/SUN
- FileMaker (Macintosh)
- FPX format
- Framework
- Framework II
- Freehand 11
- FTP Session Data
- GEM Bit Image
- Ghost Disk Image
- Google SketchUp

- Graphics Environment Manager (GEM VDI)
- Harvard Graphics
- Hewlett Packard
- Honey Bull DSA101
- HP Graphics Language (HP-GL)
- HP Graphics Language (Plotter)
- HP PCL and PJL Languages
- HP Word PC
- IBM 1403 Line Printer
- IBM DCA-FFT
- IBM DCF Script
- Informix SmartWare II
- Informix SmartWare II Communication File
- Informix SmartWare II Database
- Informix SmartWare Spreadsheet
- Interleaf
- ISO 10303-21 STEP format
- Java Class file
- JPEG File Interchange Format (JFIF)
- Keyhole Markup Language
- KW ODA G4 (G4)
- KW ODA G31D (G31)
- KW ODA Internal G32D (G32)
- KW ODA Internal Raw Bitmap (RBM)
- Lasergraphics Language
- Link Library UNIX/VAX/SUN
- Lotus Notes Bitmap
- Lotus Notes CDF
- Lotus Screen Cam
- Lyrix
- Macromedia Director
- MacWrite
- MacWrite II
- MASS-11
- MATLAB MAT Format
- Micrografx Designer
- Microsoft Access 2007
- Microsoft Access 2007 Template

- Microsoft Common Object File Format (COFF)
- Microsoft Compiled HTML Help
- Microsoft Device Independent Bitmap
- Microsoft Document Imaging (MDI)
- Microsoft Excel 2007 Macro-Enabled Spreadsheet Template
- Microsoft Excel 2007 Spreadsheet Template
- Microsoft Exchange Server Database File
- Microsoft Object File Library
- Microsoft Office Drawing
- Microsoft Office Groove
- Microsoft Outlook Restricted Permission Message File
- Microsoft Windows Cursor (CUR) Graphics
- Microsoft Windows Group File
- Microsoft Windows Help File
- Microsoft Windows Icon (ICO)
- Microsoft Windows NT Event Log
- Microsoft Windows OLE 2 Encapsulation
- Microsoft Windows Vista Event Log
- Microsoft Word (UNIX)
- Microsoft Works (Macintosh)
- Microsoft Works Communication (Macintosh)
- Microsoft Works Communication (Windows)
- Microsoft Works Database (Macintosh)
- Microsoft Works Database (PC)
- Microsoft Works Database (Windows)
- Microsoft Works Spreadsheet (Macintosh)
- Microstation
- Milestone Document
- MORE Database Outliner (Macintosh)
- MPEG4 (ISO IEC MPEG4)
- MPEG-PS container with CDXA stream
- MS DOS Batch File format
- MS DOS Device Driver
- MultiMate 4.0
- Multiplan Spreadsheet
- Navy DIF
- NBI Async Archive Format
- NBI Net Archive Format

- Nero Encrypted File
- Netscape Bookmark file
- NeWS font file (SUN)
- NIOS TOP
- Nota Bene
- NURSTOR Drawing
- Object Module UNIX/VAX/SUN
- ODA/ODIF
- ODA/ODIF (FOD 26)
- Office Writer
- OLE DIB object
- OLIDIF
- Open PGP (new format packets)
- OS/2 PM Metafile Graphics
- PaperPort image file
- Paradox (PC) Database
- PC COM executable (detected in file mode only)
- PC Library Module
- PC Object Module
- PC True Type Font
- PCD Image
- PeachCalc Spreadsheet
- Persuasion Presentation
- PEX Binary Archive (SUN)
- PGP Compressed Data
- PGP Encrypted Data
- PGP Public Keyring
- PGP Secret Keyring
- PGP Signature Certificate
- PGP Signed and Encrypted Data
- PGP Signed Data
- Philips Script
- PKCS #12 (p12) Format
- Plan Perfect
- Portable Bitmap Utilities (PBM)
- Portable Greymap Utilities (PGM)
- Portable Pixmap Utilities (PPM)
- PostScript File

- PostScript Type 1 Font File
- PRIMEWORD
- Program Information File
- PTC Creo
- Q & A for DOS
- Q & A for Windows
- Quadratron Q-One (V1.93J)
- Quadratron Q-One (V2.0)
- Quark Xpress (Macintosh)
- QuickDraw 3D Metafile (3DMF)
- Real Audio
- RealLegal E-Transcript
- Reflex Database (R2D)
- RIFF Device Independent Bitmap
- RIFF MIDI
- RIFF Multimedia Movie
- SAMNA Word IV
- Samsung Electronics JungUm Global format
- SEG-Y Seismic Data format
- Serialized Object Format (SOF) Encapsulation
- SGML
- Simple Vector Format (SVF)
- SMTP document
- SolidWorks
- Sony WAVE64 format
- Star Office Calc Spreadsheet (versions 3-5)
- Star Office Impress Presentation (versions 3-5)
- Star Office Math (versions 3-5)
- Star Office Writer Text (versions 3-5)
- StuffIt Archive (Macintosh)
- SUN vfont definition
- SYLK Spreadsheet
- Symphony Spreadsheet
- Targon Word (V 2.0)
- Unigraphics NX
- Uniplex (V6.01)
- UNIX SHAR Encapsulation
- Usenet format

- Volkswriter
- Vorbis OGG format
- VRML
- VRML 2.0
- WANG PC
- Wang WITA
- WANG WPS Comm.
- Web ARChive (WARC)
- Windows C++ Object Storage
- Windows Journal
- Windows Micrografx Draw (DRW)
- Windows Palette
- Windows scrap file (SHS)
- Wireless Markup Language
- Word Connection
- WordMARC word processor
- WordPerfect General File
- WordStar
- WordStar 6.0
- WordStar 2000
- WriteNow
- Writing Assistant word processor
- X Bitmap (XBM)
- X Image
- X Pixmap (XPM)
- Xerox 860 Comm.
- Xerox DocuWorks
- Xerox Writer word processor
- Yahoo! Messenger chat log
- Zipped Keyhole Markup Language

Appendix B: KeyView Format Codes

This section lists the KeyView format classes and codes used with Connector Framework Server.

- KeyView Classes, below
- KeyView Formats, on the next page

KeyView Classes lists KeyView file classes. The numbers are reported in the `DocumentClass` field in documents processed by CFS. Consult the table to determine the file class that was imported.

KeyView Formats, on the next page lists all KeyView formats. The numbers are reported in the `DocumentType` field in documents processed by Connector Framework Server. Consult the table to determine the file type that was imported.

You can use any of the format numbers from KeyView Formats, on the next page in conjunction with the `ImportFamilyRootExcludeFmtCSV` parameter. For more information about this parameter, refer to the *Connector Framework Server Reference*.

KeyView Classes

KeyView classes

Attribute number	File class
0	No file class
01	Word processor
02	Spreadsheet
03	Database
04	Raster image
05	Vector graphic
06	Presentation
07	Executable
08	Encapsulation
09	Sound
10	Desktop publishing
11	Outline/planning
12	Miscellaneous
13	Mixed format

KeyView classes, continued

Attribute number	File class
14	Font
15	Time scheduling
16	Communications
17	Object module
18	Library module
19	Fax
20	Movie
21	Animation

KeyView Formats

The following table lists KeyView file format codes and the file extensions they are most commonly associated with.

NOTE:
This table is not a complete list of file extensions. KeyView returns format codes based on file content, which cannot always be predicted from the file extension. Some file extensions may also be associated with multiple format numbers.

KeyView file formats and extensions

Format Name	Format Number	Format Description	Associated File Extension
AES_Multiplus_Comm_Fmt	1	Multiplus (AES)	PTF
ASCII_Text_Fmt	2	Text	
MSDOS_Batch_File_Fmt	3	MS-DOS Batch File	BAT
Applix_Alis_Fmt	4	APPLIX ASTERIX	AX
BMP_Fmt	5	Windows Bitmap	BMP
CT_DEF_Fmt	6	Convergent Technologies DEF Comm. Format	
Corel_Draw_Fmt	7	Corel Draw	CDR
CGM_ClearText_	8	Computer Graphics Metafile (CGM)	CGM[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Fmt			
CGM_Binary_Fmt	9	Computer Graphics Metafile (CGM)	CGM [1]
CGM_Character_Fmt	10	Computer Graphics Metafile (CGM)	CGM [1]
Word_Connection_Fmt	11	Word Connection	CN
COMET_TOP_Word_Fmt	12	COMET TOP	
CEOwrite_Fmt	13	CEOwrite	CW
DSA101_Fmt	14	DSA101 (Honeywell Bull)	
DCA_RFT_Fmt	15	DCA-RFT (IBM Revisable Form)	RFT
CDA_DDIF_Fmt	16	CDA / DDIF	
DG_CDS_Fmt	17	DG Common Data Stream (CDS)	CDS
Micrografx_Draw_Fmt	18	Windows Draw (Micrografx)	DRW
Data_Point_VistaWord_Fmt	19	Vistaword	
DECdx_Fmt	20	DECdx	DX
Enable_WP_Fmt	21	Enable Word Processing	WPF
EPSF_Fmt	22	Encapsulated PostScript	EPS [1]
Preview_EPSF_Fmt	23	Encapsulated PostScript	EPS [1]
MS_Executable_Fmt	24	MSDOS/Windows Program	EXE
G31D_Fmt	25	CCITT G3 1D	
GIF_87a_Fmt	26	Graphics Interchange Format (GIF87a)	GIF [1]
GIF_89a_Fmt	27	Graphics Interchange Format (GIF89a)	GIF [1]
HP_Word_PC_Fmt	28	HP Word PC	HW
IBM_1403_LinePrinter_Fmt	29	IBM 1403 Line Printer	I4

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
IBM_DCF_Script_Fmt	30	DCF Script	IC
IBM_DCA_FFT_Fmt	31	DCA-FFT (IBM Final Form)	IF
Interleaf_Fmt	32	Interleaf	
GEM_Image_Fmt	33	GEM Bit Image	IMG
IBM_Display_Write_Fmt	34	Display Write	IP
Sun_Raster_Fmt	35	Sun Raster	RAS
Ami_Pro_Fmt	36	Lotus Ami Pro	SAM
Ami_Pro_StyleSheet_Fmt	37	Lotus Ami Pro Style Sheet	
MORE_Fmt	38	MORE Database MAC	
Lyrix_Fmt	39	Lyrix Word Processing	
MASS_11_Fmt	40	MASS-11	M1
MacPaint_Fmt	41	MacPaint	PNTG
MS_Word_Mac_Fmt	42	Microsoft Word for Macintosh	DOC [1]
SmartWare_II_Comm_Fmt	43	SmartWare II	
MS_Word_Win_Fmt	44	Microsoft Word for Windows	DOC [1]
Multimate_Fmt	45	MultiMate	MM [1]
Multimate_Fnote_Fmt	46	MultiMate Footnote File	FNX [1]
Multimate_Adv_Fmt	47	MultiMate Advantage	
Multimate_Adv_Fnote_Fmt	48	MultiMate Advantage Footnote File	
Multimate_Adv_II_Fmt	49	MultiMate Advantage II	MM[1]
Multimate_Adv_II_Fnote_Fmt	50	MultiMate Advantage II Footnote File	FNX [1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Multiplan_PC_Fmt	51	Multiplan (PC)	
Multiplan_Mac_Fmt	52	Multiplan (Mac)	
MS_RTF_Fmt	53	Rich Text Format (RTF)	RTF
MS_Word_PC_Fmt	54	Microsoft Word for PC	DOC [1]
MS_Word_PC_StyleSheet_Fmt	55	Microsoft Word for PC Style Sheet	DOC [1]
MS_Word_PC_Glossary_Fmt	56	Microsoft Word for PC Glossary	DOC [1]
MS_Word_PC_Driver_Fmt	57	Microsoft Word for PC Driver	DOC [1]
MS_Word_PC_Misc_Fmt	58	Microsoft Word for PC Miscellaneous File	DOC [1]
NBI_Async_Archive_Fmt	59	NBI Async Archive Format	
Navy_DIF_Fmt	60	Navy DIF	ND
NBI_Net_Archive_Fmt	61	NBI Net Archive Format	NN
NIOS_TOP_Fmt	62	NIOS TOP	
FileMaker_Mac_Fmt	63	Filemaker MAC	FP5, FP7
ODA_Q1_11_Fmt	64	ODA / ODIF	OD[1]
ODA_Q1_12_Fmt	65	ODA / ODIF	OD [1]
OLIDIF_Fmt	66	OLIDIF (Olivetti)	
Office_Writer_Fmt	67	Office Writer	OW
PC_Paintbrush_Fmt	68	PC Paintbrush Graphics (PCX)	PCX
CPT_Comm_Fmt	69	CPT	
Lotus_PIC_Fmt	70	Lotus PIC	PIC
Mac_PICT_Fmt	71	QuickDraw Picture	PCT
Philips_Script_Word_Fmt	72	Philips Script	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
PostScript_Fmt	73	PostScript	PS
PRIMEWORD_Fmt	74	PRIMEWORD	
Quadratron_Q_One_v1_Fmt	75	Q-One V1.93J	Q1 [1], QX [1]
Quadratron_Q_One_v2_Fmt	76	Q-One V2.0	Q1 [1], QX [1]
SAMNA_Word_IV_Fmt	77	SAMNA Word	SAM
Ami_Pro_Draw_Fmt	78	Lotus Ami Pro Draw	SDW
SYLK_Spreadsheet_Fmt	79	SYLK	
SmartWare_II_WP_Fmt	80	SmartWare II	
Symphony_Fmt	81	Symphony	WR1
Targa_Fmt	82	Targa	TGA
TIFF_Fmt	83	TIFF	TIF, TIFF
Targon_Word_Fmt	84	Targon Word	TW
Uniplex_Ucalc_Fmt	85	Uniplex Ucalc	SS
Uniplex_WP_Fmt	86	Uniplex	UP
MS_Word_UNIX_Fmt	87	Microsoft Word UNIX	DOC[1]
WANG_PC_Fmt	88	WANG PC	
WordERA_Fmt	89	WordERA	
WANG_WPS_Comm_Fmt	90	WANG WPS	WF
WordPerfect_Mac_Fmt	91	WordPerfect MAC	WPM, WPD[1]
WordPerfect_Fmt	92	WordPerfect	WO, WPD[1]
WordPerfect_VAX_Fmt	93	WordPerfect VAX	WPD[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
WordPerfect_Macro_Fmt	94	WordPerfect Macro	
WordPerfect_Dictionary_Fmt	95	WordPerfect Spelling Dictionary	
WordPerfect_Thesaurus_Fmt	96	WordPerfect Thesaurus	
WordPerfect_Resource_Fmt	97	WordPerfect Resource File	
WordPerfect_Driver_Fmt	98	WordPerfect Driver	
WordPerfect_Cfg_Fmt	99	WordPerfect Configuration File	
WordPerfect_Hyphenation_Fmt	100	WordPerfect Hyphenation Dictionary	
WordPerfect_Misc_Fmt	101	WordPerfect Miscellaneous File	WPD[1]
WordMARC_Fmt	102	WordMARC	WM, PW
Windows_Metafile_Fmt	103	Windows Metafile	WMF[1]
Windows_Metafile_NoHdr_Fmt	104	Windows Metafile (no header)	WMF[1]
SmartWare_II_DB_Fmt	105	SmartWare II	
WordPerfect_Graphics_Fmt	106	WordPerfect Graphics	WPG, QPG
WordStar_Fmt	107	WordStar	WS
WANG_WITA_Fmt	108	WANG WITA	WT
Xerox_860_Comm_Fmt	109	Xerox 860	
Xerox_Writer_Fmt	110	Xerox Writer	
DIF_SpreadSheet_Fmt	111	Data Interchange Format (DIF)	DIF

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Enable_ Spreadsheet_Fmt	112	Enable Spreadsheet	SSF
SuperCalc_Fmt	113	Supercalc	CAL
UltraCalc_Fmt	114	UltraCalc	
SmartWare_II_SS_ Fmt	115	SmartWare II	
SOF_Encapsulation_ Fmt	116	Serialized Object Format (SOF)	SOF
PowerPoint_Win_ Fmt	117	PowerPoint PC	PPT[1]
PowerPoint_Mac_ Fmt	118	PowerPoint MAC	PPT[1]
PowerPoint_95_Fmt	119	PowerPoint 95	PPT[1]
PowerPoint_97_Fmt	120	PowerPoint 97	PPT[1]
PageMaker_Mac_ Fmt	121	PageMaker for Macintosh	
PageMaker_Win_ Fmt	122	PageMaker for Windows	
MS_Works_Mac_ WP_Fmt	123	Microsoft Works for MAC	
MS_Works_Mac_ DB_Fmt	124	Microsoft Works for MAC	
MS_Works_Mac_ SS_Fmt	125	Microsoft Works for MAC	
MS_Works_Mac_ Comm_Fmt	126	Microsoft Works for MAC	
MS_Works_DOS_ WP_Fmt	127	Microsoft Works for DOS	WPS[1]
MS_Works_DOS_ DB_Fmt	128	Microsoft Works for DOS	WDB[1]
MS_Works_DOS_ SS_Fmt	129	Microsoft Works for DOS	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Works_Win_WP_Fmt	130	Microsoft Works for Windows	WPS[1]
MS_Works_Win_DB_Fmt	131	Microsoft Works for Windows	WDB[1]
MS_Works_Win_SS_Fmt	132	Microsoft Works for Windows	S30, S40
PC_Library_Fmt	133	DOS/Windows Object Library	
MacWrite_Fmt	134	MacWrite	
MacWrite_II_Fmt	135	MacWrite II	
Freehand_Fmt	136	Freehand MAC	
Disk_Doubler_Fmt	137	Disk Doubler	
HP_GL_Fmt	138	HP Graphics Language	HPGL
FrameMaker_Fmt	139	FrameMaker	FM, FRM
FrameMaker_Book_Fmt	140	FrameMaker	BOOK
Maker_Markup_Language_Fmt	141	Maker Markup Language	
Maker_Interchange_Fmt	142	Maker Interchange Format (MIF)	MIF
JPEG_File_Interchange_Fmt	143	Interchange Format	JPG, JPEG
Reflex_Fmt	144	Reflex	
Framework_Fmt	145	Framework	
Framework_II_Fmt	146	Framework II	FW3
Paradox_Fmt	147	Paradox	DB
MS_Windows_Write_Fmt	148	Windows Write	WRI
Quattro_Pro_DOS_Fmt	149	Quattro Pro for DOS	
Quattro_Pro_Win_Fmt	150	Quattro Pro for Windows	WB2, WB3

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Persuasion_Fmt	151	Persuasion	
Windows_Icon_Fmt	152	Windows Icon Format	ICO
Windows_Cursor_Fmt	153	Windows Cursor	CUR
MS_Project_Activity_Fmt	154	Microsoft Project	MPP[1]
MS_Project_Resource_Fmt	155	Microsoft Project	MPP[1]
MS_Project_Calc_Fmt	156	Microsoft Project	MPP[1]
PKZIP_Fmt	157	ZIP Archive	ZIP
Quark_Xpress_Fmt	158	Quark Xpress MAC	
ARC_PAK_Archive_Fmt	159	PAK/ARC Archive	ARC, PAK
MS_Publisher_Fmt	160	Microsoft Publisher	PUB[1]
PlanPerfect_Fmt	161	PlanPerfect	
WordPerfect_Auxiliary_Fmt	162	WordPerfect auxiliary file	WPW
MS_WAVE_Audio_Fmt	163	Microsoft Wave	WAV
MIDI_Audio_Fmt	164	MIDI	MID, MIDI
AutoCAD_DXF_Binary_Fmt	165	AutoCAD DXF	DXF[1]
AutoCAD_DXF_Text_Fmt	166	AutoCAD DXF	DXF[1]
dBase_Fmt	167	dBase	DBF
OS_2_PM_Metafile_Fmt	168	OS/2 PM Metafile	MET
Lasergraphics_Language_Fmt	169	Lasergraphics Language	
AutoShade_	170	AutoShade Rendering	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Rendering_Fmt			
GEM_VDI_Fmt	171	GEM VDI	VDI
Windows_Help_Fmt	172	Windows Help File	HLP
Volkswriter_Fmt	173	Volkswriter	VW4
Ability_WP_Fmt	174	Ability	
Ability_DB_Fmt	175	Ability	
Ability_SS_Fmt	176	Ability	
Ability_Comm_Fmt	177	Ability	
Ability_Image_Fmt	178	Ability	
XyWrite_Fmt	179	XYWrite / Nota Bene	XY4
CSV_Fmt	180	CSV (Comma Separated Values)	CSV
IBM_Writing_ Assistant_Fmt	181	IBM Writing Assistant	IWA
WordStar_2000_Fmt	182	WordStar 2000	WS2
HP_PCL_Fmt	183	HP Printer Control Language	PCL
UNIX_Exe_ PreSysV_VAX_Fmt	184	Unix Executable (PDP-11/pre-System V VAX)	
UNIX_Exe_Basic_ 16_Fmt	185	Unix Executable (Basic-16)	
UNIX_Exe_x86_Fmt	186	Unix Executable (x86)	
UNIX_Exe_iAPX_ 286_Fmt	187	Unix Executable (iAPX 286)	
UNIX_Exe_MC68k_ Fmt	188	Unix Executable (MC680x0)	
UNIX_Exe_3B20_ Fmt	189	Unix Executable (3B20)	
UNIX_Exe_ WE32000_Fmt	190	Unix Executable (WE32000)	
UNIX_Exe_VAX_ Fmt	191	Unix Executable (VAX)	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
UNIX_Exe_Bell_5_Fmt	192	Unix Executable (Bell 5.0)	
UNIX_Obj_VAX_Demand_Fmt	193	Unix Object Module (VAX Demand)	
UNIX_Obj_MS8086_Fmt	194	Unix Object Module (old MS 8086)	
UNIX_Obj_Z8000_Fmt	195	Unix Object Module (Z8000)	
AU_Audio_Fmt	196	NeXT/Sun Audio Data	AU
NeWS_Font_Fmt	197	NeWS bitmap font	
cpio_Archive_CRChdr_Fmt	198	cpio archive (CRC Header)	
cpio_Archive_CHRhdr_Fmt	199	cpio archive (CHR Header)	
PEX_Binary_Archive_Fmt	200	SUN PEX Binary Archive	
Sun_vfont_Fmt	201	SUN vfont Definition	
Curses_Screen_Fmt	202	Curses Screen Image	
UUEncoded_Fmt	203	UU encoded	UUE
WriteNow_Fmt	204	WriteNow MAC	
PC_Obj_Fmt	205	DOS/Windows Object Module	
Windows_Group_Fmt	206	Windows Group	
TrueType_Font_Fmt	207	TrueType Font	TTF
Windows_PIF_Fmt	208	Program Information File (PIF)	PIF
MS_COM_Executable_Fmt	209	PC (.COM)	COM
StuffIt_Fmt	210	StuffIt (MAC)	HQX
PeachCalc_Fmt	211	PeachCalc	
Wang_GDL_Fmt	212	WANG Office GDL Header	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Q_A_DOS_Fmt	213	Q & A for DOS	
Q_A_Win_Fmt	214	Q & A for Windows	JW
WPS_PLUS_Fmt	215	WPS-PLUS	WPL
DCX_Fmt	216	DCX FAX Format(PCX images	DCX
OLE_Fmt	217	OLE Compound Document	OLE
EBCDIC_Fmt	218	EBCDIC Text	
DCS_Fmt	219	DCS	
UNIX_SHAR_Fmt	220	SHAR	SHAR
Lotus_Notes_ BitMap_Fmt	221	Lotus Notes Bitmap	
Lotus_Notes_CDF_ Fmt	222	Lotus Notes CDF	CDF
Compress_Fmt	223	Unix Compress	Z
GZ_Compress_Fmt	224	GZ Compress	GZ[1]
TAR_Fmt	225	TAR	TAR
ODIF_FOD26_Fmt	226	ODA / ODIF	F26
ODIF_FOD36_Fmt	227	ODA / ODIF	F36
ALIS_Fmt	228	ALIS	
Envoy_Fmt	229	Envoy	EVY
PDF_Fmt	230	Portable Document Format	PDF
BinHex_Fmt	231	BinHex	HQX
SMTP_Fmt	232	SMTP	SMTP
MIME_Fmt	233	MIME[2]	EML, MBX
USENET_Fmt	234	USENET	
SGML_Fmt	235	SGML	SGML
HTML_Fmt	236	HTML	HTM[1], HTML [1]
ACT_Fmt	237	ACT	ACT

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
PNG_Fmt	238	Portable Network Graphics (PNG)	PNG
MS_Video_Fmt	239	Video for Windows (AVI)	AVI
Windows_Animated_Cursor_Fmt	240	Windows Animated Cursor	ANI
Windows_CPP_Obj_Storage_Fmt	241	Windows C++ Object Storage	
Windows_Palette_Fmt	242	Windows Palette	PAL
RIFF_DIB_Fmt	243	RIFF Device Independent Bitmap	
RIFF_MIDI_Fmt	244	RIFF MIDI	RMI
RIFF_Multimedia_Movie_Fmt	245	RIFF Multimedia Movie	
MPEG_Fmt	246	MPEG Movie	MPG, MPEG[1]
QuickTime_Fmt	247	QuickTime Movie, MPEG-4 Audio	MOV, QT, MP4
AIFF_Fmt	248	Audio Interchange File Format (AIFF)	AIF, AIFF
Amiga_MOD_Fmt	249	Amiga MOD	MOD
Amiga_IFF_8SVX_Fmt	250	Amiga IFF (8SVX) Sound	IFF
Creative_Voice_Audio_Fmt	251	Creative Voice (VOC)	VOC
AutoDesk_Animator_FLI_Fmt	252	AutoDesk Animator FLIC	FLI
AutoDesk_AnimatorPro_FLC_Fmt	253	AutoDesk Animator Pro FLIC	FLC
Compactor_Archive_Fmt	254	Compactor / Compact Pro	
VRML_Fmt	255	VRML	WRL
QuickDraw_3D_Metafile_Fmt	256	QuickDraw 3D Metafile	
PGP_Secret_	257	PGP Secret Keyring	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Keyring_Fmt			
PGP_Public_ Keyring_Fmt	258	PGP Public Keyring	
PGP_Encrypted_ Data_Fmt	259	PGP Encrypted Data	
PGP_Signed_Data_ Fmt	260	PGP Signed Data	
PGP_ SignedEncrypted_ Data_Fmt	261	PGP Signed and Encrypted Data	
PGP_Sign_ Certificate_Fmt	262	PGP Signature Certificate	
PGP_Compressed_ Data_Fmt	263	PGP Compressed Data	
PGP_ASCII_Public_ Keyring_Fmt	264	ASCII-armored PGP Public Keyring	
PGP_ASCII_ Encoded_Fmt	265	ASCII-armored PGP encoded	PGP[1]
PGP_ASCII_ Signed_Fmt	266	ASCII-armored PGP encoded	PGP[1]
OLE_DIB_Fmt	267	OLE DIB object	
SGI_Image_Fmt	268	SGI Image	RGB
Lotus_ScreenCam_ Fmt	269	Lotus ScreenCam	
MPEG_Audio_Fmt	270	MPEG Audio	MPEGA
FTP_Software_ Session_Fmt	271	FTP Session Data	STE
Netscape_ Bookmark_File_Fmt	272	Netscape Bookmark File	HTM[1]
Corel_Draw_CMX_ Fmt	273	Corel CMX	CMX
AutoDesk_DWG_ Fmt	274	AutoDesk Drawing (DWG)	DWG

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
AutoDesk_WHIP_Fmt	275	AutoDesk WHIP	WHP
Macromedia_Director_Fmt	276	Macromedia Director	DCR
Real_Audio_Fmt	277	Real Audio	RM
MSDOS_Device_Driver_Fmt	278	MSDOS Device Driver	SYS
Micrografx_Designer_Fmt	279	Micrografx Designer	DSF
SVF_Fmt	280	Simple Vector Format (SVF)	SVF
Applix_Words_Fmt	281	Applix Words	AW
Applix_Graphics_Fmt	282	Applix Graphics	AG
MS_Access_Fmt	283	Microsoft Access	MDB[1]
MS_Access_95_Fmt	284	Microsoft Access 95	MDB[1]
MS_Access_97_Fmt	285	Microsoft Access 97	MDB[1]
MacBinary_Fmt	286	MacBinary	BIN
Apple_Single_Fmt	287	Apple Single	
Apple_Double_Fmt	288	Apple Double	
Enhanced_Metafile_Fmt	289	Enhanced Metafile	EMF
MS_Office_Drawing_Fmt	290	Microsoft Office Drawing	
XML_Fmt	291	XML	XML[1]
DeVice_Independent_Fmt	292	DeVice Independent file (DVI)	DVI
Unicode_Fmt	293	Unicode	UNI
Lotus_123_Worksheet_Fmt	294	Lotus 1-2-3	WK1[1]
Lotus_123_Format_Fmt	295	Lotus 1-2-3 Formatting	FM3

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Lotus_123_97_Fmt	296	Lotus 1-2-3 97	WK1[1]
Lotus_Word_Pro_96_Fmt	297	Lotus Word Pro 96	LWP[1]
Lotus_Word_Pro_97_Fmt	298	Lotus Word Pro 97	LWP[1]
Freelance_DOS_Fmt	299	Lotus Freelance for DOS	
Freelance_Win_Fmt	300	Lotus Freelance for Windows	PRE
Freelance_OS2_Fmt	301	Lotus Freelance for OS/2	PRS
Freelance_96_Fmt	302	Lotus Freelance 96	PRZ[1]
Freelance_97_Fmt	303	Lotus Freelance 97	PRZ[1]
MS_Word_95_Fmt	304	Microsoft Word 95	DOC[1]
MS_Word_97_Fmt	305	Microsoft Word 97	>DOC[1]
Excel_Fmt	306	Microsoft Excel	XLS[1]
Excel_Chart_Fmt	307	Microsoft Excel	XLS[1]
Excel_Macro_Fmt	308	Microsoft Excel	XLS[1]
Excel_95_Fmt	309	Microsoft Excel 95	XLS[1]
Excel_97_Fmt	310	Microsoft Excel 97	XLS[1]
Corel_Presentations_Fmt	311	Corel Presentations	XFD, XFDL
Harvard_Graphics_Fmt	312	Harvard Graphics	
Harvard_Graphics_Chart_Fmt	313	Harvard Graphics Chart	CH3, CHT
Harvard_Graphics_Symbol_Fmt	314	Harvard Graphics Symbol File	SY3
Harvard_Graphics_Cfg_Fmt	315	Harvard Graphics Configuration File	
Harvard_Graphics_Palette_Fmt	316	Harvard Graphics Palette	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Lotus_123_R9_Fmt	317	Lotus 1-2-3 Release 9	
Applix_ Spreadsheets_Fmt	318	Applix Spreadsheets	AS
MS_Pocket_Word_ Fmt	319	Microsoft Pocket Word	PWD, DOC[1]
MS_DIB_Fmt	320	MS Windows Device Independent Bitmap	
MS_Word_2000_Fmt	321	Microsoft Word 2000	DOC[1]
Excel_2000_Fmt	322	Microsoft Excel 2000	XLS[1]
PowerPoint_2000_ Fmt	323	Microsoft PowerPoint 2000	PPT
MS_Access_2000_ Fmt	324	Microsoft Access 2000	MDB[1], MPP[1]
MS_Project_4_Fmt	325	Microsoft Project 4	MPP[1]
MS_Project_41_Fmt	326	Microsoft Project 4.1	MPP[1]
MS_Project_98_Fmt	327	Microsoft Project 98	MPP[1]
Folio_Flat_Fmt	328	Folio Flat File	FFF
HWP_Fmt	329	HWP(Arae-Ah Hangul)	HWP
ICHITARO_Fmt	330	ICHITARO V4-10	
IS_XML_Fmt	331	Extended or Custom XML	XML[1]
Oasys_Fmt	332	Oasys format	OA2, OA3
PBM_ASC_Fmt	333	Portable Bitmap Utilities ASCII Format	
PBM_BIN_Fmt	334	Portable Bitmap Utilities Binary Format	
PGM_ASC_Fmt	335	Portable Greymap Utilities ASCII Format	
PGM_BIN_Fmt	336	Portable Greymap Utilities Binary Format	PGM
PPM_ASC_Fmt	337	Portable Pixmap Utilities ASCII	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
		Format	
PPM_BIN_Fmt	338	Portable Pixmap Utilities Binary Format	
XBM_Fmt	339	X Bitmap Format	XBM
XPM_Fmt	340	X Pixmap Format	XPM
FPX_Fmt	341	FPX Format	FPX
PCD_Fmt	342	PCD Format	PCD
MS_Visio_Fmt	343	Microsoft Visio	VSD
MS_Project_2000_ Fmt	344	Microsoft Project 2000	MPP[1]
MS_Outlook_Fmt	345	Microsoft Outlook	MSG, OFT
ELF_Relocatable_ Fmt	346	ELF Relocatable	O
ELF_Executable_ Fmt	347	ELF Executable	
ELF_Dynamic_Lib_ Fmt	348	ELF Dynamic Library	SO
MS_Word_XML_Fmt	349	Microsoft Word 2003 XML	XML[1]
MS_Excel_XML_Fmt	350	Microsoft Excel 2003 XML	XML[1]
MS_Visio_XML_Fmt	351	Microsoft Visio 2003 XML	VDX
SO_Text_XML_Fmt	352	StarOffice Text XML	SXW[1], ODT[1]
SO_Spreadsheet_ XML_Fmt	353	StarOffice Spreadsheet XML	SXC[1], ODS[1]
SO_Presentation_ XML_Fmt	354	StarOffice Presentation XML	SXI[1], SXP[1], ODP[1]
XHTML_Fmt	355	XHTML	XML[1]
MS_OutlookPST_ Fmt	356	Microsoft Outlook PST	PST
RAR_Fmt	357	RAR	RAR
Lotus_Notes_NSF_	358	IBM Lotus Notes Database NSF/NTF	NSF

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Fmt			
Macromedia_Flash_Fmt	359	SWF	SWF
MS_Word_2007_Fmt	360	Microsoft Word 2007 XML	DOCX, DOTX
MS_Excel_2007_Fmt	361	Microsoft Excel 2007 XML	XLSX, XLTX
MS_PPT_2007_Fmt	362	Microsoft PPT 2007 XML	PPTX, POTX, PPSX
OpenPGP_Fmt	363	OpenPGP Message Format (with new packet format)	PGP
Intergraph_V7_DGN_Fmt	364	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)	DGN[1]
MicroStation_V8_DGN_Fmt	365	MicroStation V8 DGN (OLE)	DGN[1]
MS_Word_Macro_2007_Fmt	366	Microsoft Word Macro 2007 XML	DOCM, DOTM
MS_Excel_Macro_2007_Fmt	367	Microsoft Excel Macro 2007 XML	XLSM, XLTM, XLAM
MS_PPT_Macro_2007_Fmt	368	Microsoft PPT Macro 2007 XML	PPTM, POTM, PPSM, PPAM
LZH_Fmt	369	LHA Archive	LZH, LHA
Office_2007_Fmt	370	Office 2007 document	XLSB
MS_XPS_Fmt	371	Microsoft XML Paper Specification (XPS)	XPS
Lotus_Domino_DXL_Fmt	372	IBM Lotus representation of Domino design elements in XML format	DXL
ODF_Text_Fmt	373	ODF Text	ODT[1], SXW[1], STW
ODF_Spreadsheet_Fmt	374	ODF Spreadsheet	ODS[1], SXC[1], STC
ODF_Presentation_Fmt	375	ODF Presentation	SXD[1], SXI[1], ODG[1], ODP[1]
Legato_Extender_ONM_Fmt	376	Legato Extender Native Message ONM	ONM

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
bin_Unknown_Fmt	377	n/a	
TNEF_Fmt	378	Transport Neutral Encapsulation Format (TNEF)	various
CADAM_Drawing_Fmt	379	CADAM Drawing	CDD
CADAM_Drawing_Overlay_Fmt	380	CADAM Drawing Overlay	CDO
NURSTOR_Drawing_Fmt	381	NURSTOR Drawing	NUR
HP_GLP_Fmt	382	HP Graphics Language (Plotter)	HPG
ASF_Fmt	383	Advanced Systems Format (ASF)	ASF
WMA_Fmt	384	Window Media Audio Format (WMA)	WMA
WMV_Fmt	385	Window Media Video Format (WMV)	WMV
EMX_Fmt	386	Legato EMailXtender Archives Format (EMX)	EMX
Z7Z_Fmt	387	7 Zip Format(7z)	7Z
MS_Excel_Binary_2007_Fmt	388	Microsoft Excel Binary 2007	XLSB
CAB_Fmt	389	Microsoft Cabinet File (CAB)	CAB
CATIA_Fmt	390	CATIA Formats (CAT*)	CAT[3]
YIM_Fmt	391	Yahoo Instant Messenger History	DAT[1]
ODF_Drawing_Fmt	392	ODF Drawing	SXD[1], SX[1], ODG[1]
Founder_CEB_Fmt	393	Founder Chinese E-paper Basic (ceb)	CEB
QPW_Fmt	394	Quattro Pro 9+ for Windows	QPW
MHT_Fmt	395	MHT format[2]	MHT
MDI_Fmt	396	Microsoft Document Imaging Format	MDI
GRV_Fmt	397	Microsoft Office Groove Format	GRV
IWWP_Fmt	398	Apple iWork Pages format	PAGES, GZ[1]
IWSS_Fmt	399	Apple iWork Numbers format	NUMBERS, GZ[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
IWPG_Fmt	400	Apple iWork Keynote format	KEY, GZ[1]
BKF_Fmt	401	Windows Backup File	BKF
MS_Access_2007_ Fmt	402	Microsoft Access 2007	ACCDB
ENT_Fmt	403	Microsoft Entourage Database Format	
DMG_Fmt	404	Mac Disk Copy Disk Image File	
CWK_Fmt	405	AppleWorks File	
OO3_Fmt	406	Omni Outliner File	OO3
OPML_Fmt	407	Omni Outliner File	OPML
Omni_Graffle_XML_ Fmt	408	Omni Graffle XML File	GRAFFLE
PSD_Fmt	409	Photoshop Document	PSD
Apple_Binary_PList_ Fmt	410	Apple Binary Property List format	
Apple_iChat_Fmt	411	Apple iChat format	
OOUTLINE_Fmt	412	OOutliner File	OOUTLINE
BZIP2_Fmt	413	Bzip 2 Compressed File	BZ2
ISO_Fmt	414	ISO-9660 CD Disc Image Format	ISO
DocuWorks_Fmt	415	DocuWorks Format	XDW
RealMedia_Fmt	416	RealMedia Streaming Media	RM, RA
AC3Audio_Fmt	417	AC3 Audio File Format	AC3
NEF_Fmt	418	Nero Encrypted File	NEF
SolidWorks_Fmt	419	SolidWorks Format Files	SLDASM, SLDPRT, SLDDRW
XFDL_Fmt	420	Extensible Forms Description Language	XFDL, XFD
Apple_XML_PList_ Fmt	421	Apple XML Property List format	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
OneNote_Fmt	422	OneNote Note Format	ONE
Dicom_Fmt	424	Digital Imaging and Communications in Medicine	DCM
EnCase_Fmt	425	Expert Witness Compression Format (EnCase)	E01, L01, Lx01
Scrap_Fmt	426	Shell Scrap Object File	SHS
MS_Project_2007_Fmt	427	Microsoft Project 2007	MPP[1]
MS_Publisher_98_Fmt	428	Microsoft Publisher 98/2000/2002/2003/2007/	PUB[1]
Skype_Fmt	429	Skype Log File	DBB
Hl7_Fmt	430	Health level7 message	HL7
MS_OutlookOST_Fmt	431	Microsoft Outlook OST	OST
Epub_Fmt	432	Electronic Publication	EPUB
MS_OEDBX_Fmt	433	Microsoft Outlook Express DBX	DBX
BB_Activ_Fmt	434	BlackBerry Activation File	DAT[1]
DiskImage_Fmt	435	Disk Image	
Milestone_Fmt	436	Milestone Document	MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9
E_Transcript_Fmt	437	RealLegal E-Transcript File	PTX
PostScript_Font_Fmt	438	PostScript Type 1 Font	PFB
Ghost_DiskImage_Fmt	439	Ghost Disk Image File	GHO, GHS
JPEG_2000_JP2_File_Fmt	440	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	JP2, JPF, J2K, JPWL, JPX, PGX
Unicode_HTML_Fmt	441	Unicode HTML	HTM[1], HTML[1]
CHM_Fmt	442	Microsoft Compiled HTML Help	CHM
EMCMF_Fmt	443	Documentum EMCMF format	EMCMF
MS_Access_2007_	444	Microsoft Access 2007 Template	ACCDT

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Tmpl_Fmt			
Jungum_Fmt	445	Samsung Electronics Jungum Global document	GUL
JBIG2_Fmt	446	JBIG2 File Format	JB2, JBIG2
EFax_Fmt	447	eFax file	EFX
AD1_Fmt	448	AD1 Evidence file	AD1
SketchUp_Fmt	449	Google SketchUp	SKP
GWFS_Email_Fmt	450	Group Wise File Surf email	GWFS
JNT_Fmt	451	Windows Journal format	JNT
Yahoo_yChat_Fmt	452	Yahoo! Messenger chat log	YCHAT
PaperPort_MAX_File_Fmt	453	PaperPort image file	MAX
ARJ_Fmt	454	ARJ (Archive by Robert Jung) file format	ARJ
RPMSG_Fmt	455	Microsoft Outlook Restricted Permission Message	RPMSG
MAT_Fmt	456	MATLAB file format	MAT, FIG
SGY_Fmt	457	SEG-Y Seismic Data format	SGY, SEGY
CDXA_MPEG_PS_Fmt	458	MPEG-PS container with CDXA stream	MPG[1]
EVT_Fmt	459	Microsoft Windows NT Event Log	EVT
EVTX_Fmt	460	Microsoft Windows Vista Event Log	EVTX
MS_OutlookOLM_Fmt	461	Microsoft Outlook for Macintosh format	OLM
WARC_Fmt	462	Web ARChive	WARC
JAVACLASS_Fmt	463	Java Class format	CLASS
VCF_Fmt	464	Microsoft Outlook vCard file format	VCF
EDB_Fmt	465	Microsoft Exchange Server Database file format	EDB

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
ICS_Fmt	466	Microsoft Outlook iCalendar file format	ICS, VCS
MS_Visio_2013_Fmt	467	Microsoft Visio 2013	VSDX, VSTX, VSSX
MS_Visio_2013_ Macro_Fmt	468	Microsoft Visio 2013 macro	VSDM, VSTM, VSSM
ICHITARO_Compr_ Fmt	469	ICHITARO Compressed format	JTDC
IWWP13_Fmt	470	Apple iWork 2013 Pages format	IWA
IWSS13_Fmt	471	Apple iWork 2013 Numbers format	IWA
IWPG13_Fmt	472	Apple iWork 2013 Keynote format	IWA
XZ_Fmt	473	XZ archive format	XZ
Sony_WAVE64_Fmt	474	Sony Wave64 format	W64
Conifer_WAVPACK_ Fmt	475	Conifer Wavpack format	WV
Xiph_OGG_ VORBIS_Fmt	476	Xiph Ogg Vorbis format	OGG
MS_Visio_2013_ Stencil_Fmt	477	MS Visio 2013 stencil format	VSSX
MS_Visio_2013_ Stencil_Macro_Fmt	478	MS Visio 2013 stencil Macro format	VSSM
MS_Visio_2013_ Template_Fmt	479	MS Visio 2013 template format	VSTX
MS_Visio_2013_ Template_Macro_ Fmt	480	MS Visio 2013 template Macro format	VSTM
Borland_Reflex_2_ Fmt	481	Borland Reflex 2 format	R2D
PKCS_12_Fmt	482	PKCS #12 (p12) format	P12, PFX
B1_Fmt	483	B1 format	B1
ISO_IEC_MPEG_4_ Fmt	484	ISO/IEC MPEG-4 format	MP4

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
RAR5_Fmt	485	RAR5 Format	RAR5
Unigraphics_NX_Fmt	486	Unigraphics (UG) NX CAD Format	PRT
PTC_Creo_Fmt	487	PTC Creo CAD Format	ASM, PRT
KML_Fmt	488	Keyhole Markup Language	KML
KMZ_Fmt	489	Zipped Keyhole Markup Language	KMZ
WML_Fmt	490	Wireless Markup Language	WML
SO_Text_Fmt	492	Star Office Writer Text	SDW, SGL, VOR
SO_Spreadsheet_Fmt	493	Star Office Calc Spreadsheet	SDC
SO_Presentation_Fmt	494	Star Office Impress Presentation	SDD, SDA
SO_Math_Fmt	495	Star Office Math	SMF
STEP_Fmt	496	ISO 10303-21 STEP format	STEP
STL_Fmt	497	3D Systems STL format	STL
MS_Word_2007_Flat_XML_Fmt	546	Microsoft Word 2007 Flat XML	XML

[1]This file extension can return more than one format number.

[2]MHT, EML, and MBX files might return either format 2, 233, or 395, depending on the text in the file. In general, files that contain fields such as **To**, **From**, **Date**, or **Subject** are considered to be email messages; files that contain fields such as **content-type** and **mime-version** are considered to be MHT files; and files that do not contain any of those fields are considered to be text files.

[3]All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

Appendix C: Document Fields

This appendix describes the standard fields that Connectors and CFS add to documents before the documents are indexed into IDOL Server.

Document Fields

The following fields are added to a document by connectors:

Field	Description
`AUTN_IDENTIFIER`	An identifier that allows a connector to extract the document from the repository again, for example during the collect or view actions. For more information about the identifier, see AUTN_ IDENTIFIER, on the next page.
`DocTrackingId`	An identifier used for document tracking functionality.
`DREREFERENCE`	A reference for the document. This is the standard IDOL reference field, which is used for deduplication.
`source_connector_run_id`	(Added only when `IngestSourceConnectorFields=TRUE`). The asynchronous action token of the `fetch` action that ingested the document.
`source_connector_server_id`	(Added only when `IngestSourceConnectorFields=TRUE`). A token that identifies the instance of the connector that retrieved the document (different installations of the same connector populate this field with different IDs). You can retrieve the UID of a connector through `action=GetVersion`.

The following fields are added to a document during ingestion:

Field	Description
`DocumentAttributes`	KeyView document attributes.
`DocumentClass`	The KeyView document class.
`DocumentSize`	The size of the document.
`DocumentType`	A number that represents the program that created the file format.
`DRECHILDCOUNT`	The number of sub-files that the document contains.

Field	Description
`DREDBNAME`	The name of the IDOL database that the document must be indexed to.
`DREFILENAME`	The file name of the original document.
`DREORIGINALNAME`	The original file name passed to CFS. This is the full path for extracted sub-files.
`DREROOTFAMILYREFERENCE`	The parent document for the family of documents.
`DREROOTFAMILYREFERENCE_ID`	A unique hash for the family of documents.
`FAMILYSORT`	A field used to track families (that is, containers) of documents. It contains a hash unique to the family, with indices appended that describe the depth and number of attachments.
`ImportErrorDescription`	If an error occurs when a document is processed, a description of the error is written to this field.
`ImportMagicExtension`	The file extension of the detected document type.
`ImportOriginalEncoding`	The detected encoding used by the document.
`ImportVersion`	Internal version number.
`InfoFlag`	A KeyView Flag that describes the file type (External, Embedded and so on). 0 = default 1 = This sub file needs further extraction 2 = This sub file is protected 4 = This sub file is an external file 8 = This sub file is a mail item attachment 16 = This sub file is SMIME protected
`KeyviewVersion`	The version of KeyView that CFS was released with.
`UUID`	A unique identifier for the document.
`VersionNumber`	The version of CFS that was used to import the document.

AUTN_IDENTIFIER

An *Identifier* is a base-64 encoded string that identifies the source of a document in IDOL Server. When you use a connector to index documents into IDOL Server, an identifier is added to every document, in the `AUTN_IDENTIFIER` document field.

A connector can use the identifier to extract the original file from the repository. An application might need to extract the original file when presenting the results of a query. The application can request the file by sending a `collect` or `view` action to the connector.

The exact content of the `AUTN_IDENTIFIER` field depends on the connector that retrieved the document, but contains information such as:

- The document reference. The document reference identifies an item in a repository. For the files retrieved from the same repository, a reference is unique. For files retrieved by a File System Connector, the document reference is the path to the file. For e-mail messages retrieved by an Exchange Connector, the document reference includes the name of the message store and folder that contains the message.
- Additional information used to find the document in the repository. Though the document reference identifies a file in the repository, it might not provide sufficient information to retrieve it efficiently. The identifier can include additional information to assist the connector locate the document.
- The name of the fetch task that was used to retrieve the document. When a connector needs to retrieve a file, it can use the same settings by finding the fetch task in its configuration file.

An example identifier appears below:

```
<id section="MyTask1" reference="http://myserver:4567/doc/_vxswdfguhjknbio_
earycqzt_">
  <param name="SERVICEURL" value="http://myserver:4567/service"/>
  <param name="DOCID">_vxswdfguhjknbio_earycqzt_</param>
</id>
```

Sub File Indexes

Documents in IDOL Server can represent sub-files. In these documents, the `AUTN_IDENTIFIER` field contains the identifier of the container file.

To retrieve a sub-file from a repository, a connector must retrieve the container file and send it to KeyView so that the sub-file can be extracted. So that KeyView can extract the correct sub-file, the identifier must include a sub-file index.

When CFS indexes documents into IDOL Server, sub-file indexes are automatically written to the `SubFileIndexCSV` document field. For example:

```
SubFileIndexCSV="1"
```

> **NOTE:**
> Your connector must be configured with `EnableExtraction=true`. The connector's `KeyviewDirectory` parameter must also be set.

The sub-file index in this example (`1`) indicates that the document represents the second file in the container (the sub-files are indexed from 0).

Container files can contain other container files (for example an e-mail message file could contain ZIP file attachments, containing further sub-files). In this case, the sub-file index might include more than one level:

```
SubFileIndexCSV="2,6"
```

A sub-file index of `2,6` indicates that the document represents the seventh file in the third container, in the original container file.

When an action is sent to a connector to retrieve sub-files, the sub-file index must be appended to the identifier of the container. For example:

```
PGlkIHM9Ik15VGFzazEiIHI9Imh0dHA6Ly9teXNlcnZlcjo0NTY3L2RvYy9fdnhzd
2RmZ3VoamtuYmlvX2VhcnljcXp0XyI+PHAgbj0iU0VSVklDRVVSTCIgdj0iaHR0cD
ovL215c2VydmVyOjQ1Njcvc2VydmljZSIvPjxwIG49IkRPQ0lEIiB2PSJfdnhzd2R
mZ3VoamtuYmlvX2VhcnljcXp0XyIvPjwvaWQ+|2.6
```

> **NOTE:**
> Where sub-file indexes have multiple levels (for example `SubFileIndexCSV="2,6"`, the comma must be replaced by a period).

Append Sub File Indexes to the Document Identifier

You can configure CFS to automatically append sub-file indexes to document identifiers, before the documents are indexed into IDOL Server.

To do this, use the lua script `identifiers.lua`, which is included with CFS in the `scripts` folder. The script is also included below:

```
function handler( document )
   identifier = document:getFieldValue( "AUTN_IDENTIFIER" )

   if identifier then
      indices = document:getFieldValue( "SubFileIndexCSV" )

      if indices then
         indices = string.gsub(indices, ",", ".")
         document:setFieldValue("AUTN_IDENTIFIER", identifier .. "|" .. indices)
      end
   end

   return true
end
```

You must run the script after KeyView has extracted sub-files, so run the script using a *Post* Import task. For example:

```
[ImportTasks]
Post0=Lua:scripts/identifiers.lua
```

Glossary

A

ACI (Autonomy Content Infrastructure)
A technology layer that automates operations on unstructured information for cross-enterprise applications. ACI enables an automated and compatible business-to-business, peer-to-peer infrastructure. The ACI allows enterprise applications to understand and process content that exists in unstructured formats, such as email, Web pages, Microsoft Office documents, and IBM Notes.

ACI Server
A server component that runs on the Autonomy Content Infrastructure (ACI).

ACL (access control list)
An ACL is metadata associated with a document that defines which users and groups are permitted to access the document.

action
A request sent to an ACI server.

active directory
A domain controller for the Microsoft Windows operating system, which uses LDAP to authenticate users and computers on a network.

C

Category component
The IDOL Server component that manages categorization and clustering.

Community component
The IDOL Server component that manages users and communities.

connector
An IDOL component (for example File System Connector) that retrieves information from a local or remote repository (for example, a file system, database, or Web site).

Connector Framework Server (CFS)
Connector Framework Server processes the information that is retrieved by connectors. Connector Framework Server uses KeyView to extract document content and metadata from over 1,000 different file types. When the information has been processed, it is sent to an IDOL Server or Distributed Index Handler (DIH).

Content component
The IDOL Server component that manages the data index and performs most of the search and retrieval operations from the index.

D

DAH (Distributed Action Handler)
DAH distributes actions to multiple copies of IDOL Server or a component. It allows you to use failover, load balancing, or distributed content.

DIH (Distributed Index Handler)
DIH allows you to efficiently split and index extremely large quantities of data into multiple copies of IDOL Server or the Content component. DIH allows you to create a scalable solution that delivers high performance and high availability. It provides a flexible way to batch, route, and categorize the indexing of internal and external content into IDOL Server.

I

IDOL

The Intelligent Data Operating Layer (IDOL) Server, which integrates unstructured, semi-structured and structured information from multiple repositories through an understanding of the content. It delivers a real-time environment in which operations across applications and content are automated.

IDOL Proxy component

An IDOL Server component that accepts incoming actions and distributes them to the appropriate subcomponent. IDOL Proxy also performs some maintenance operations to make sure that the subcomponents are running, and to start and stop them when necessary.

Intellectual Asset Protection System (IAS)

An integrated security solution to protect your data. At the front end, authentication checks that users are allowed to access the system that contains the result data. At the back end, entitlement checking and authentication combine to ensure that query results contain only documents that the user is allowed to see, from repositories that the user has permission to access. For more information, refer to the IDOL Document Security Administration Guide.

K

KeyView

The IDOL component that extracts data, including text, metadata, and subfiles from over 1,000 different file types. KeyView can also convert documents to HTML format for viewing in a Web browser.

L

LDAP

Lightweight Directory Access Protocol. Applications can use LDAP to retrieve information from a server. LDAP is used for directory services (such as corporate email and telephone directories) and user authentication. See also: active directory, primary domain controller.

License Server

License Server enables you to license and run multiple IDOL solutions. You must have a License Server on a machine with a known, static IP address.

O

OmniGroupServer (OGS)

A server that manages access permissions for your users. It communicates with your repositories and IDOL Server to apply access permissions to documents.

P

primary domain controller

A server computer in a Microsoft Windows domain that controls various computer resources. See also: active directory, LDAP.

V

View

An IDOL component that converts files in a repository to HTML formats for viewing in a Web browser.

W

Wildcard

A character that stands in for any character or group of characters in a query.

X

XML

Extensible Markup Language. XML is a language that defines the different attributes of document content in a format that can be read by humans and machines. In IDOL Server, you can index documents in XML format. IDOL Server also returns action responses in XML format.

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Administration Guide (Micro Focus Connector Framework Server 11.6)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.idoldocsfeedback@microfocus.com.

We appreciate your feedback!