Import module

version 2.5.x - revision 3

Administrator's

Guide

Table of Contents

Autonomy

Autonomy employs a fundamentally different and unique combination of technologies to enable computers to form an understanding of a page of text, web pages, emails, voice, documents and people.

Autonomy's solution is therefore able to power any application dependent upon unstructured information within every market sector, including: e-commerce, customer relationship management, knowledge management, enterprise information portals and online publishing applications.

This is evidenced by the significant penetration of the technology in a diversity of vertical markets and has been achieved principally because every market sector needs to manage and leverage the benefits of unstructured information.

Autonomy was founded in 1996 and has offices in Boston, Chicago, Dallas, San Francisco, New York, and Washington, D.C. in the United States, as well as offices throughout EMEA, including Amsterdam, Brussels, Cambridge, Frankfurt, Milan, Paris, Oslo, and Sydney. In July 1998, the company went public on the EASDAQ exchange (EASDAQ:AUTN). Autonomy floated on The NASDAQ National Market (NASDAQ: AUTN) in May 2000, and on the London Stock Exchange (LSE: AU.) in November 2000.

To contact Autonomy, please get in touch with your nearest location listed below.

Europe and South Pacific

Autonomy Systems Ltd.
Cambridge Business Park
Cowley Road
Cambridge
CB4 0WZ

Help Desk:	+44 (0) 800 0 282 858	
Switchboard:	+44 (0) 1223 448 000	
Fax:	+44 (0) 1223 448 001	
Email	for information:	autonomy@autonomy.com
	for support:	uksupport@autonomy.com

The Help Desk operates from 9.30 am to 6.00 pm (GMT) Monday to Friday.
Website: www.autonomy.com

USA

Autonomy Inc.
301 Howard Street
22nd Floor
San Francisco
CA 94105

Help Desk:	+1 877 333 7744	
Switchboard:	+1 415 243 9955	
Fax:	+1 415 243 9984	
Email	for information:	info@us.autonomy.com
	for support:	support@us.autonomy.com

The Help Desk operates from 9.30 am to 6.00 pm (CST) Monday to Friday, toll-free.
Website: www.autonomy.com

Welcome

Thank you for choosing Autonomy and welcome to your Import module version 2.5 Administrator's Guide.

Autonomy Solutions

Autonomy solutions provide the software infrastructure that automates operations on unstructured information. This software infrastructure is based on IDOL server, the Intelligent Data Operating Layer. IDOL makes it possible for organizations to process digital content automatically and to enable applications to operate with each other. It consists of data operations that integrate information by understanding any type of content, and is therefore data agnostic. The IDOL server software infrastructure is fully scalable and customizable according to customers' present and future needs.

Autonomy solutions include:

- **Autonomy Connectors ™** enable automatic content aggregation from any type of local or remote repository (for example, a database, a web site, a real-time telephone conversation etc.), facilitating a unified solution across all information assets within the organization.
- **ACI Servers ™** automatically perform a variety of operations on structured, unstructured and semi-structured information (documents, audio, video etc.). These operations include automatic hyperlinking, tag reconciliation, XML tagging, profiling, alerting, categorization, cluster mapping, real-time transcription, targeting etc.
- **Autonomy Application Builder ™** is a toolkit that enables companies and partners to customize Autonomy's products according to their individual requirements. It facilitates easy communication between custom-built applications that retrieve data using HTTP commands and the Autonomy ACI servers, as well as simple manipulation of the returned result sets. Communication with the servers is implemented over HTTP using XML and can adhere to SOAP. The API is distributed with a set of sample code.
- **Portal-in-a-Box™**, our comprehensive and fully automated Information Portal for content-rich Internet and Intranet sites.
- **Portlets** are windows that can be set up in Autonomy's Portal-in-a-Box or third party Portals. Each portlet contains an application that allows the Portals' end users to benefit from a variety of IDOL server functionality.
- **Autonomy Desktop Suite™** brings the power of Autonomy to every desktop. Conducting a real-time analysis of the ideas involved in the content of any opened desktop application, Desktop Suite's ActiveKnowledge or Active Windows Extensions module provides real-time links to relevant internal and external information without the user being needlessly diverted from their work in progress to perform an exasperating search or retrieval operation.
- **Autonomy Product Orientated Drop-in Solutions ™** allow Autonomy solutions to be easily integrated with third party applications and solution providers. PODS enable organizations to make their existing applications compatible with IDOL with minimal configuration and administration requirements. Making IDOL server a part of any solution delivers the direct benefits of content automation and the ability to perform a vast range of IDOL server operations, irrelevant of file format or location.

1. Introduction

The Import module is the importing process that allows you to import different document types into IDX or XML file format, so they can be indexed into IDOL server.

The Import module is integral to Autonomy connectors (for example, HTTPFetch, Oracle fetch and so on), which retrieve data from different sources automatically, use the Import module to import it into IDX or XML file format (in order to make it IDOL server compatible) and index it into the IDOL server.

The Import module comprises the following components:

- importslavex
- pdfslave
- omnislave
- wpconvdll.dll
- wordconv.dll
- rtfconv.dll
- excelconv.dll
- pptconv.dll
- binslave
- pstconv.dll

Supported formats

The Import module allows you to import the following data formats. Note that some audio and visual formats require additional plugins, which are available from Autonomy.

Word processing

- HTML
- SGML
- XML
- TEXT
- RTF
- WML
- Adobe PDF
- ASCII text
- ANSI text
- Unicode V2.x
- Microsoft RTF
- Microsoft Word for Windows V3.x onwards
- Microsoft Word Mac V4.x to 6.x
- Microsoft Word PC V2.0 to 5.5
- Quark QXD

Spreadsheet

- Microsoft Works V3.x onwards
- Microsoft Excel V3.x onwards.

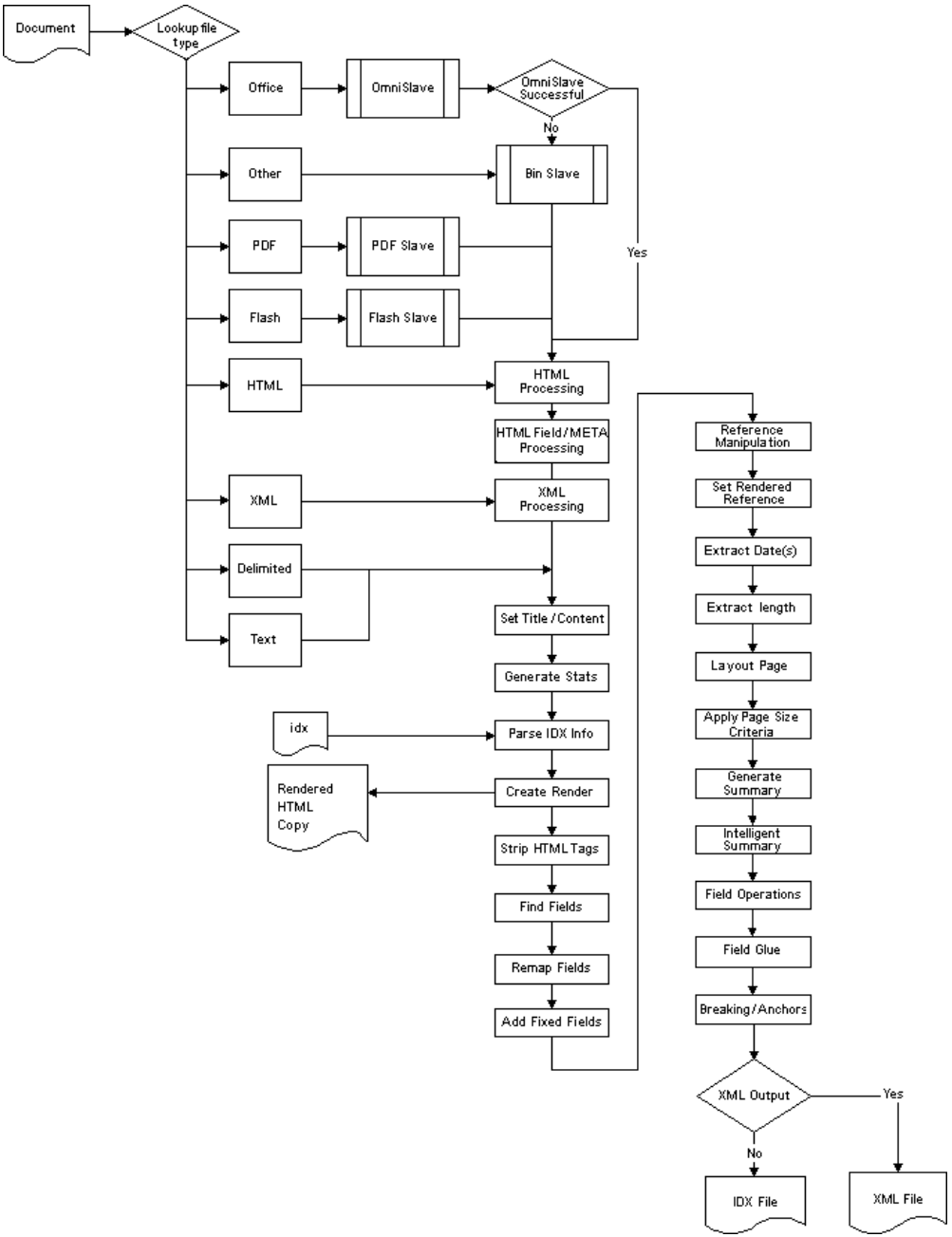
Presentation Graphics Formats

- Shockwave Flash (with Autonomy Flashslave)
- Microsoft Powerpoint 4 onwards

Audio (with Autonomy VoiceSuite)

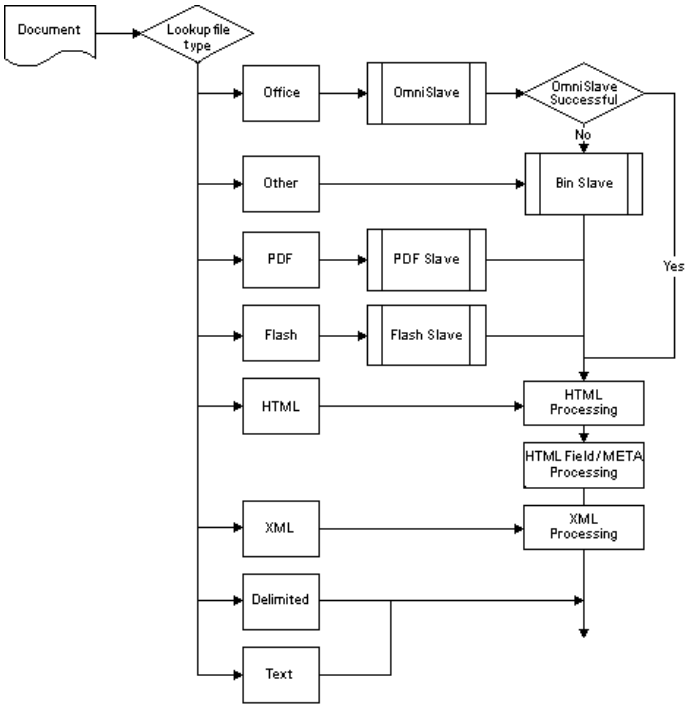
- Windows Media Audio (WMA) and Windows Media Audio metafiles (WAX)
- Windows Media Video (WMV) and Windows Media Video metafiles (WVX)
- Windows Media files (ASF) and Windows Media metafiles (ASX)
- Video On Demand (VOD)
- Moving Picture Experts Group standard 1 and 2 (MPEG-1, MPEG-2), including MPEG Layer-3 audio(MP3)
- Closed captioning (SMI, SAMI)
- Non-streaming (local playback) formats supported by the Windows Media
- Compact Disc Audio (CD)
- Digital Video Disc (DVD)
- Audio-Video Interleaved (AVI)
- Apple QuickTime® (QT, MOV), version 2 and lower
- IndeoT Video 5
- Waveform Audio (WAV)
- Sound File (SND)
- UNIX audio (AU)
- Audio Interchange File Format (AIFF)
- RM, RA, RAM RealAudio/RealVideo streamed content
- RT RealText streamed text formats
- MP3 MPEG Layer 3 (audio format)
- SWF RealSystem G2 with Flash
- SMIL, SMI SMIL files (see SMIL)
- WAV, AIFF* 'Legacy' sound files-older but prevalent file types
- MPG, MPEG Standard MPEG Layer 1 video and Layer 2 audio formats
- AVI* Audio/Video Interleave-Microsoft video format
- ASF* NetShow files

System architecture



Importing process

File type identification



The Import module identifies the file type of a document using its file extension. According to the file type it does the following:

Microsoft Office files

The Import module uses Omnislave to convert the file into a temporary HTML file. If it does not succeed in converting the file (for example, if there are problems with the formatting), it passes the file on to binslave. Omnislave always extracts metadata from the files.

PDF files

The Import module uses the PDFslave to convert the file into a temporary HTML file.

Flash files

The Import module uses the Flashslave to convert the file into a temporary HTML file.

HTML or XML files

The Import module identifies the section from which data should be imported. From XML files it extracts all data content by default. From HTML files it also extracts the HTML fields and meta tags.

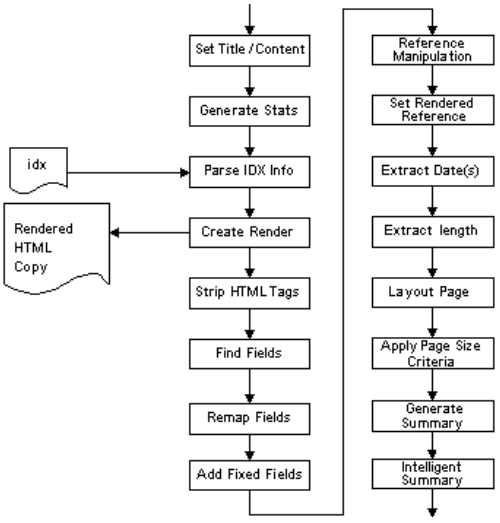
Plain text or delimited plain text files

The Import module immediately extracts data.

Other files

The Import module uses Bin Slave to convert the file into a temporary HTML file. Note that Bin Slave is not able to extract the metadata within binary headers.

Data extraction

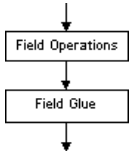


1. The Import module extracts the title and content of the document.
2. The Import module generates fields containing the statistics for the document for example, the length of the document in bytes, the number of words that the document contains and so on).
3. If a stub IDX file exists for the document, the Import module extracts fields and values from the stub IDX file.

   Note: a stub IDX file contains meta fields and values that overwrite fields and values from the original document or that are added to the file that is imported into IDOL server.
4. A rendered HTML copy of the document is saved in the location specified for **ImportRenderedHTMLMoveToDir**.
5. The Import module strips the HTML tags from the data.
6. The Import module identifies where fields start and end and extracts the text from these fields.

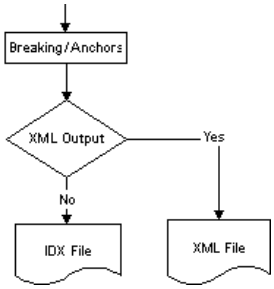
7. If fixed fields have been set in the Import parameters, the Import module adds these to the fields from the document.
8. The Import module replaces part of the IDOL server reference so that the document's URL can be accessed.
9. If an HTML copy of the document has been created, the Import module sets the IDOL server reference to this rendered copy.
10. The Import module extracts the length of the file (in bytes).
11. The Import module extracts text for importing.
12. If **ImportBreaking** has been enabled, the Import module breaks up the document according to the values specified for **ImportBreakingMinDocWords**, **ImportBreakingMinParagraphWords** and **ImportBreakingMaxParagraphWords**.

Field mapping



1. The Import module creates a summary field with content taken from the quick summary generated by the Import module.
2. If **ImportIntelligentSummary** has been enabled, the Import module compares the document with other imported documents and ensures that it has a unique title and summary.
3. The Import module performs field operations that have been specified in the Import parameters (for example, stripping or replacing particular characters from fields).
4. The Import module glues fields together as specified in the Import parameters.

Importing



1. The Import module breaks the document into subsections.
2. The Import module imports the extracted data and fields into an IDX or XML file.

2. Configuring the Import module

You can specify import parameters in a connector's configuration file in order to determine how the Import module operates.

Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

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

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

Entering string values

If the value that you want to enter for a parameter that requires a string contains quotation marks, you must put the value into quotation marks and escape each quotation mark that the string contains by putting a slash in front of it.

For example:

FIELDSTART0="<font face=\"arial\"size=\"+1\"><b>"

Here the beginning and end of the string is indicated by quotation marks while all quotation marks that are contained in the string are escaped.

If you want to enter a comma separated list of strings for a parameter, and one of the strings contains a comma, you must indicate the start and the end of this string with quotation marks.

For example:

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

If any string within a comma separated list contains quotation marks, you must put this string into quotation marks and escaped the quotation marks in the string by putting a slash in front of them.

For example:

ParameterName="<font face=\"arial\"size=\"+1\"><b>",dog,bird,"wing,beak",turtle

Applying modifications to a connector's operation

New configuration settings only take effect once the connector service is stopped and restarted.

Generic settings

ImportTempDir

Enter the full path to the directory in which the Import module can store the temporary files that it produces when it imports non-HTML/Text documents, converts data from one character set to another and so on.

ImportTempDIr must always be valid for the Import module to operate successfully. By default the Import module uses the current directory to store temporary files.

ImportPoliteness

Specify the amount of time (in milliseconds) to wait between documents that you are importing.

For example:

ImportPoliteness=1000

In this example, the Import module pauses for 1 second (that is, 1000 milliseconds) after importing a document, before it imports the next document.

The default value for this setting is **0**.

ImportPreImportMaxLength

Specify the maximum size (in bytes) of files that can be imported. The Import module checks file size before the file is processed, and does not continue with processing if the file size is greater than **ImportPreImportMaxLength**.

You can enter **-1** if you don't want the Import module to check the file size before importing. (This is the default value for this setting.)

Storing content in IDOL server

The following settings allow you to specify where and how content is stored in the IDOL server.

Database

Allows you to specify the name of the IDOL server database into which documents are indexed when they are imported.

ImportDocumentType

When a document that has been indexed into IDOL server is returned as a result in a front-end application, you can display an icon alongside it in order to indicate the type or source of the document. **Note:** this is only possible if your front-end application supports this feature and you have set up appropriate fields.

Enter an extension to indicate the type of documents that you are importing.

For example:

ImportDocumentType=txt

ImportStoreContent

Note: this setting is supported for backwards compatibility only. If you are importing into IDOL server or a DRE version 4 or higher you do not need to set **ImportStoreContent** as content is always stored.

Enter **true** if you want to store the plain text content of imported documents in the IDOL server (this is the default setting). This allows you to view the content of an imported document even when its source does not exist anymore. **Note:** if at some point you want to create a backup IDX file of the IDOL server contents you should set **ImportStoreContent** to **true**.

Enter **false** if you do not want to store plain text content in the IDOL server. This option allows you to save disk space, however, if an imported document is removed from its source you will not be able to access it anymore.

Importing to XML

The following settings allow you to import documents into XML rather than IDX format. You can then use the XML for your own applications or index it into IDOL server.

ImportXMLOutput

Enter **true** if you want to import retrieved documents into XML file format.

Enter **false** if you want to import retrieved documents into IDX file format. This is the default.

ImportXMLOutputHeader

If you have set **ImportXMLOutput** to **true**, you can enter a string to specify a header for the XML output.

For example:

ImportXMLOutputHeader=<?xml version="1.0" encoding="iso-8859-1"?>

ImportXMLOutputDocumentDelimiter

If you have set **ImportXMLOutput** to **true**, you can specify a delimiter that is added to the start and end of each document that is included in the output XML file. The default value is **DREDOC**.

For example:

ImportXMLOutputDocumentDelimiter=XMLDOC

ImportXMLOutputRootTagName

If you have set **ImportXMLOutput** to **true**, you can specify the name of the root tag in the XML that is output. By default this is **DOCS**.

Selecting documents for importing

The following settings allow you to specify which documents in the import directory are eligible for importing into IDOL server. You can also use them to filter out poor quality content.

ImportMinLength

Allows you to specify the minimum size that a document must have (in bytes) in order to be imported. The document size is measured by its plain text content. By default this is **0**.

For example:

ImportMinLength=500

In this example only documents that have a size of at least **500** bytes can be imported.

ImportMaxLength

Allows you to specify the maximum size that a document can have (in bytes) in order to be imported. The document size is measured by its plain text content. By default this is **10000000**.

For example:

ImportMaxLength=10000000

In this example only documents that have a size of **10000000** bytes or less can be imported.

ImportMinLengthWords

Allows you to specify the minimum number of words that a document must contain in order to be imported. By default this is **0**.

For example:

ImportMinLengthWords=200

In this example no document that contains less than **200** words can be imported.

ImportMaxLengthWords

Allows you to specify the maximum number of words that a document is permitted to contain in order to be imported. By default this is **1000000**.

For example:

ImportMaxLengthWords=500

In this example no document that contains more than **500** words can be imported.

ImportMaxAnchors

The maximum number of anchors in an HTML document that can be imported. Once the maximum number of anchors has been imported the importing process of the HTML document is stopped, and the rest of the document is discarded.

By default a maximum of **2048** anchors is imported.

ImportMaxPlainTextLengthKBs

When you are importing text files you can use **ImportMaxPlainTextLengthKBs** to discard files that are larger than the specified size. Enter a number to specify the maximum size in kilobytes that a plain text document is permitted to have.

Only documents whose size is less that the specified size will be imported.

ImportRecursiveDirectoryImporting

Enter **true** if you want to import not only files that are contained in the import directory but also any files that are contained in subdirectories that the import directory may contain. This is the default.

Enter **false** if you only want to import files that are contained in the import directory.

Note: the directory from which files are imported is specified in the individual connector's configuration file.

ImportMustHaveCSVs

One or more strings that a document must contain in order to be imported. Use **ImportMustHaveCheck** to specify which part of a document must contain the specified strings.

If you want to specify multiple strings you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings that you specify.

For example:

ImportMustHaveCheck=1

ImportMustHaveCSVs=*PDF*,*pdf*

In this example the Import module checks if documents contain the strings **PDF** or **pdf** in their IDOL server reference (as part of a word or whole word). If this is not the case, the document is discarded.

Note: you can use **ImportMustHaveFieldCSVs** to specify additional document fields to check in for the strings.

ImportMustHaveFieldCSVs

Enter one or more fields that you want to check for the strings you specify with **ImportMustHaveCSVs**.

If you want to specify multiple field names, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportMustHaveCheck=64

ImportMustHaveFieldCSVs=TITLE,THEME

ImportMustHaveCSVs=*france*,*travel*

In this example, the Import module performs a case-insensitive check for whether documents contain the strings **france** or **travel** in their **TITLE** or **THEME** fields. If a document doesn't contain either of the strings in one of the fields, the document is discarded.

Note: one of the fields that you specify with **ImportMustHaveFieldCSVs** can be **DREREFERENCE**.

ImportMustHaveCheck

Allows you to specify how the **ImportMustHaveCSVs** strings should be applied by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

Check specified fields: 0

If you enter **0** the Import module checks the fields you specify with **ImportMustHaveFieldCSVs** for any of the strings specified for **ImportMustHaveCSVs**. If none of the specified fields in a document contains any of the strings, the document is not imported.

IDOL server reference: 1

If you enter **1** the Import module checks if the IDOL server reference of a document contains any of the strings specified for **ImportMustHaveCSVs**. If it does not, the document is not imported.

Case insensitive: 64

If you add **64** to the **ImportMustHaveCheck** value, the Import module checks the fields you specify for any of the strings entered for **ImportMustHaveCSVS**, and discards the document if it does not find a case-insensitive match of any of the **ImportMustHaveCSVS** strings.

For example:

ImportMustHaveCheck=65

In this example the Import module checks the IDOL server reference of documents for case-insensitive matches of any **ImportMustHaveCSVS**. If the IDOL server reference of a document does not contain any of the **ImportMustHaveCSVS** strings, the Import module discards the document.

Notes:

- The Import module performs **CantHave** and **MustHave** checks as a last stage of the importing process, after field operations are executed. This means that if a field is modified during the importing process, it is the modified value that the Import module checks for the strings you specify with **ImportMustHaveCSVs**.
- **CantHave** requirements have precedence over **MustHave** requirements. That is, if the same string is specified for both **ImportMustHaveCSVs** and **ImportCantHaveCSVs**, and the Import module finds that string in a document, the document is not imported.

ImportCantHaveCSVs

Enter one or more strings that a document must not contain in order to be imported. Use **ImportCantHaveCheck** to specify which part of a document must not contain the specified strings.

If you want to specify multiple strings you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings that you specify.

For example:

ImportCantHaveCheck=1

ImportCantHaveCSVs=*PDF*,*pdf*

In this example the Import module checks if documents contain the strings **PDF** or **pdf** in their IDOL server reference (as part of a word or whole word). If this is the case, the document is discarded.

Note: you can use **ImportCantHaveFieldCSVs** to specify additional document fields to check in for the strings.

ImportCantHaveFieldCSVs

Enter one or more fields that you want to check for the strings you specify with **ImportCantHaveCSVs**.

If you want to specify multiple field names, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportCantHaveCheck=64

ImportCantHaveFieldCSVs=DocType,Description

ImportCantHaveCSVs=*pdf*

In this example, the Import module performs a case-insensitive check for whether documents contain the string **pdf** in their **DocType** or **Description** fields. If a document contains the string in either of the fields, the document is discarded.

Note: one of the fields that you specify with **ImportCantHaveFieldCSVs** can be **DREREFERENCE**.

ImportCantHaveCheck

Allows you to specify how the **ImportCantHaveCSVs** strings should be applied by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

Check specified fields: 0

If you enter **0**, the Import module checks the fields you specify with **ImportCantHaveFieldCSVs** for any of the strings specified for **ImportCantHaveCSVs**. If any of the fields in a document contains any of the strings, the document is not imported.

IDOL server reference: 1

If you enter **1** the Import module checks if the IDOL server reference of a document contains any of the strings specified for **ImportCantHaveCSVs**. If it does, the document is not imported.

Case insensitive: 64

If you add **64** to the **ImportCantHaveCheck** value, the Import module checks the fields you specify for any of the strings entered for **ImportCantHaveCSVS**, and discards the document if it finds a case insensitive match of any of the **ImportCantHaveCSVS** strings.

For example:

ImportMustHaveCheck=65

In this example the Import module checks the IDOL server reference of documents for case-insensitive matches of any **ImportCantHaveCSVS**. If the IDOL server reference of a document contains any of the **ImportCantHaveCSVS** strings, the Import module discards the document.

Notes:

- The Import module performs **CantHave** and **MustHave** checks as a last stage of the importing process, after field operations are executed. This means that if a field is modified during the importing process, it is the modified value that the Import module checks for the strings you specify with **ImportCantHaveCSVs**.
- **CantHave** requirements have precedence over **MustHave** requirements. That is, if the same string is specified for both **ImportMustHaveCSVs** and **ImportCantHaveCSVs**, and the Import module finds that string in a document, the document is not imported.

Import process logging

The following settings allow you to set up the way in which the Import module logs information. The logs give you details of what is being processed, the success rate, any problems encountered and how to solve them.

ImportLogFile

Enter the name of the file in which you want to log the importing process.

ImportLogFileAppend

Enter **true** if you want to append to the **ImportLogFile** log file every time the Import module cycles.

Enter **false** if you want to create a new log file every time the Import module cycles (this is the default).

ImportLogLevel

Allows you to specify which type of logging should be performed. Enter one of the following:

Warnings

An entry is made into the log file every time an error or a warning occurs.

Errors

An entry is made into the log file every time a critical error occurs.

Full

Every action and occurrence is logged. This is the default option.

None

Logging is disabled.

ImportLogFileMaxSize

Enter the maximum number of bytes that the **ImportLogFile** log file is permitted to reach before it is deleted or archived and a new log file is created.

ImportResilienceCheck

Enter **true** if you want the Import module to list the documents that it has imported successfully in a file called **ImportResilienceCheck.log**. This allows the calling application to check for failed documents.

By default this is **false**.

ImportReturnFailures

Enter **true** if you want the Import module to list details of files which are not imported successfully in a log file called **ImportFailureList.log**.

Otherwise, enter **false**. This is the default.

ImportLogExpireAction

Enter one of the following to determine how log files are treated once they reach their **ImportLogFileMaxSize** size:

Previous

The log file's name is postfixed with **.previous** and saved in the installation directory. Every time a log file reaches its **ImportLogFileMaxSize** size, it is given the same postfix, so that it overwrites the old log file. This is the default.

Datestamp

The log file's name is postfixed with a time stamp and saved in the installation directory.

Compress

The log file's name is postfixed with a time stamp and saved in the installation directory (as for **Datestamp**). The file is then compressed.

Consecutive

The log file's name is postfixed with a number and saved in the installation directory. When the next log file reaches its **ImportLogFileMaxSize** size, it is postfixed with the next consecutive number.

Identifying file types

The following settings allow you to specify how files with particular extensions are treated during the importing process.

ImportMagicEnable

Enter **true** if you want the Import module to work out the type of files that it imports automatically by inspecting the file's binary data (this is the default). This ensures that files which have an incorrect or no extension are imported correctly.

Note: if a file has no extension and the Import module cannot work out its type by looking at its binary data, the Import module assumes that the file has the extension specified for **ImportDefaultExtension**.

If you enter **false**, the Import module imports the file according to its extension. If a file has no extension, the Import module assumes that the file has the extension specified for **ImportDefaultExtension**.

ImportDefaultExtension

If during the importing process a file is found that has an extension which is not recognized, it will be treated as the file type you specify here.

Note: If you want to specify multiple extensions, you need to separate them with commas (there should be no space before or after the comma).

For example:

ImportDefaultExtension=*.txt

In this example any file whose type is not recognized is treated as a text file during the importing process

ImportDelimitedExtns

Allows you to specify one or more file extensions that should be treated as delimited files when they are imported.

Note: If you want to specify multiple extensions, you need to separate them with commas (there must be no space before or after a comma).

For example:

ImportDelimitedExtns=*.html,*.dat,*.nfo

In this example, any **html**, **dat** or **nfo** file is treated as an delimited file during the importing process.

ImportHTMLExtns

Allows you to specify one or more file extensions that should be treated as HTML when they are imported.

If you want to specify multiple extensions, you need to separate them with commas (there should be no space before or after the comma).

For example:

ImportHTMLExtns=*.htm,*.shtml,*.sgml

In this example, if a file with the extension **.htm**, **.shtml** or **.sgml** is retrieved, it will be imported as an HTML file.

ImportTextExtns

Allows you to specify one or more file extensions that should be treated as text when they are imported.

Note: If you want to specify multiple extensions, you need to separate them with commas (there should be no space before or after the comma).

For example:

ImportTextExtns=*.doc,*.txt,*.rtf

In this example, if a file with the extension **.doc**, **.txt** or **.rtf** is retrieved, it will be imported as a text file.

ImportXMLExtns

Allows you to specify one or more extensions of files that should be treated as XML files when they are imported.

Note: if you want to specify multiple extensions you need to separate them with commas (there must be no space before or after a comma).

For example:

ImportXMLExtns=*.xml,*.xhtml,*.sgml

In this example, all file that have the extension **.xml**, **.xhtml** or **.sgml** will be imported as XML files.

ImportCharsetConvExtns

Allows you to specify the extension of files which use a character set that you want to convert. Use **ImportCharsetConvEncodingFrom** and **ImportCharsetConvEncodingTo** to specify the character set that these files use and the character set that you want to convert it to.

For example:

ImportCharsetConvExtns=*.txt

ImportCharsetConvEncodingFrom=UCS2

ImportCharsetConvEncodingTo=SHIFTJIS

In this example, all files that have the extension **.txt** are converted from **UCS2** to **SHIFTJIS** before they are imported.

ImportCharsetConvEncodingFrom

Allows you to specify the character set that **ImportCharsetConvExtns** file types use. Use **ImportCharsetConvEncodingTo** to specify the character set that you want to convert this character set to.

For example:

ImportCharsetConvExtns=*.txt

ImportCharsetConvEncodingFrom=UTF8

ImportCharsetConvEncodingTo=CYRILLICKOI8

In this example, all files that have the extension **.txt** are converted from **UTF8** to **CYRILLICKOI8** before they are imported.

ImportCharsetConvEncodingTo

Allows you to specify the character set that you want to convert the **ImportCharsetConvExtns** file type's character set to. Use **ImportCharsetConvEncodingFrom** to specify the character set that you are converting from.

For example:

ImportCharsetConvExtns=*.txt

ImportCharsetConvEncodingFrom=UCS2

ImportCharsetConvEncodingTo=SHIFTJIS

In this example, all files that have the extension **.txt** are converted from **UCS2** to **SHIFTJIS** before they are imported.

ImportCharsetConvTablesDirectory

Specify the directory in which you have stored the TB*.txt files that are needed to convert the character sets that you have specified for **ImportCharsetConvEncodingFrom**. (The TB*.txt files are provided by Autonomy).

For example:

ImportCharsetConvTablesDirectory=C:\LanguageModules

ImportUnparsableExtensionCSVs

Allows you to specify one or more extensions of files that are unparseable. The Import module will not try to parse the content of these files; instead it sets the **DRECONTENT** IDOL server field with a value created from the IDOL server reference, with the following symbols replaced by a space:

. \ / > - _ ( ) &

For example:

ImportUnparsableExtensionCSVs=gif,tiff

In this example, the Import module will not attempt to parse the content of any files that have the extension **gif** or **tiff**. If the IDOL server reference for a document called **picture.gif** is C:\Autonomy\filters\picture.gif, then instead of trying to parse the content of the gif file, the Import module imports the file with the DRECONTENT field value **C Autonomy filters picture gif**.

Importing structured fields

The following settings allow you to extract and manipulate structured fields, so they can be indexed into IDOL server.

Note: the name that you give to the fields must not start with the string **DRE**. Field names that start with **DRE** are reserved for Autonomy (for example, **DRECONTENT**).

FieldName*n*

Allows you to specify the name of the field in the IDOL server configuration file in which the dynamic field value (marked by **FieldStart*n*** and **FieldStop*n***) is stored.

FieldStart*n*

Specifies the string that marks the start of the value that will be stored in the **FieldName*n*** IDOL server field.

FieldStop*n*

Specifies the string that marks the end of the value that will be stored in the **FieldName*n*** IDOL server field.

FixedFieldName*n*

Specifies the name of the fixed field in the IDOL server configuration file that will be used to store the **FixedFieldValue*n*** value.

FixedFieldValue*n*

Specifies the value that will be stored in the **FixedFieldName*n*** IDOL server field.

FixedFieldOverwriteExistingValue*n*

Enter **true** if you want the field value for the imported document to overwrite any existing value in the **FixedFieldName*n*** field. This is the default value for this setting.

Enter **false** if you want to keep the existing value in the **FixedFieldName*n*** field.

HTMLFieldName*n*

Allows you to specify the name of the field in the IDOL server configuration file in which a dynamic HTML field value (marked by **HTMLFieldStart*n*** and **HTMLFieldStop*n***) is stored. The value of the field is extracted from the HTML file, before the file is converted to plain text.

HTMLFieldStart*n*

Specifies the string that marks the start of the value that will be stored in the **HTMLFieldName*n*** IDOL server field.

HTMLFieldStop*n*

Specifies the string that marks the end of the value that will be stored in the **HTMLFieldName*n*** IDOL server field.

ImportSummary

Enter **true** to import the first sentences of documents as a summary into the IDOL server (this is the default). You can specify the number of sentences using **ImportSummarySize**.

Enter **false** if you don't require a summary.

ImportIntelligentTitleSummary

Enter **true** to generate a unique intelligent title and summary for the documents that are imported into the IDOL server. During the importing process titles and summaries that are repetitive across an import branch are deleted.

Enter **false** if you do not want to generate a unique intelligent title and summary for imported documents (this is the default).

ImportSummarySize

If you have set **ImportSummary** to **true**, you can specify the number of sentences that you want to import into the IDOL server as a document summary. The sentences are taken from the start of the document.

ImportChecksum

Enter **true** to create a Checksum field in the documents that you are importing. The Import module sets a numeric value for the Checksum field that uniquely identifies the document, based on its content. IDOL server uses this value to determine whether a document is identical to another document with a different DREREFERENCE value. Duplicate documents are discarded.

This field must be specified in the [Field] section of the IDOL server configuration file.

By default **ImportChecksum** is **false**.

ImportExtractDateDefault

Enter one of the following to determine the Import module's behavior if it cannot find a date to extract in a document:

Now

The Import module imports the document and stores the current date in the document's date field. This is the default setting.

None

The Import module imports the document without a date.

ImportExtractDateFormatCSVs*n*

For each date that you want to extract, enter one of the following to specify the format of the **ImportExtractDateFrom*n*** date that will be extracted (if you are extracting to the DREDATE field, you must enter either DD/MM/YYYY or EPOCHSECONDS):

- EPOCHSECONDS (to extract the result document's date in seconds since 1st January 1970)
- a string that contains one or more of the following:

YY	Year (2 digits), for example, 99, 00, 01 and so on
YYYY	Year (4 digits), for example, 1999, 2000, 2001 and so on
LONGMONTH	A **DateLongMonthCSVs** long month
SHORTMONTH	A **DateMonthCSVs** short month
MM	Month (2 digits), for example, 01, 10, 12 and so on
M+	Month (1 or 2 digits), for example, 1,2,3,10 and so on
DD	Day (2 digits), for example, 01, 02, 03, 12, 23 and so on
D+	Day (1 or 2 digits), for example, 1, 2, 12, 13, 31 and so on
LONGDAY	2 digits with a **DatePostFixCSVs** postfix
HH	Hour (2 digits), for example, 01, 12, 13 and so on
H+	Hour (1 or 2 digits)
NN	Minute (2 digits)
N+	Minute (1 or 2 digits)
SS	Second (2 digits)
S+	Second (1 or 2 digits)
ZZZ	Time Zone (for example, GMT, EST, PST, and so on)

Note: format strings are matched in the order in which they are listed. You should therefore put the format in order of length (starting with the longest). This prevents the Import module from matching, for example, 19/10/2002 with DD/MM/YY (if this format has been listed before DD/MM/YYYY) and extracting the date 19/10/20.

If you want to enter one or more formats, you need to separate them with commas (there must be no space before or after a comma). If you want to specify a format that contains a space, you must put the format in quotation marks. If you want to import a date that contains a comma, you must escape the comma in the date string that you set for it.

For example:

ImportExtractDateFormatCSVs0=D+/SHORTMONTH/YYYY,DDMMYY

In this example only dates that have the format **D+/SHORTMONTH/YYYY** (for example, 2/Jan/2001) or **DDMMYY** (for example, 020101) can be extracted.

ImportExtractDateFormatCSVs0="D+SHORTMONTH YYYY","Date: D+ LONGMONTH, YYYY"

In this example only dates that have the format **D+ SHORTMONTH YYYY** (for example, 2 Jan 2001) or **Date: D+ LONGMONTH, YYYY** (for example, Date: 2 January, 2001) can be extracted.

ImportExtractDateFormatCSVs0="\"LONGMONTH DD YYYY"\", DDMMYY

In this example only dates that have the format **LONGMONTH DD YYYY** (for example, November 05 2002) or **DDMMYY** (for example, 020101) can be extracted.

ImportExtractDateFrom*n*

For each date that you want to extract, enter one of the following to specify what date will be extracted from the document:

- **0** No date is extracted.
- **1** The current time is extracted.
- **2** The date that the document was last accessed is extracted.
- **4** The time that the document was created is extracted.
- **8** The date that the document was last modified is extracted.
- **16** The date is extracted from the **ImportExtractDateFromField*n*** IDOL server field.
- **32** The date is extracted from the document's content.
- **64** The date is extracted from the document's file name

For example:

ImportExtractDateFrom0=64

ImportExtractDateFromField*n*

If you have set **ImportExtractDateFrom*n*** to **16**, you can use **ImportExtractDateFromField*n*** to specify the name of the date field in the IDOL server from which you want to extract the date value.

For example:

ImportExtractDateFrom0=16

ImportExtractDateFromField0=myDateField

ImportExtractDateToField*n*

For each date that you want to extract, enter a string to specify the name of the IDOL server field in which the **ImportExtractDateFrom*n*** value is stored. Note that if you are extracting to the DREDATE field, you must set **ImportExtractDateToFormat*n*** to either YYYY/MM/DD or EPOCHSECONDS.

For example:

ImportExtractDateToField0=myDREdatefield

ImportExtractDateToFormat*n*

For each date the you want to extract, **ImportExtractDateToFormat*n*** allows you to specify in which format you want to store the **ImportExtractDateFrom*n*** date, after it has been extracted into IDOL server. You can use one of the following to specify the format in which the **ImportExtractDateFrom*n*** date will be stored:

- EPOCHSECONDS (to store the result document's date in seconds since 1st January 1970)
- a string that contains one or more of the following:

YY	Year (2 digit), for example, 99, 00, 01 and so on.
YYYY	Year (4 digit), for example, 1999, 2000, 2001 and so on.
LONGMONTH	A **DateLongMonthCSVs** long month.
SHORTMONTH	A **DateMonthCSVs** short month.
MM	Month (2 digit), for example, 01, 10, 12 and so on.
M+	Month (1 or 2 digits), for example, 1, 2, 3, 10 and so on.
DD	Day (2 digit), for example, 01, 02, 03, 12, 23 and so on.
D+	Day (1 or 2 digits), for example, 1, 2, 12, 13, 31 and so on.
LONGDAY	2 digits with a **DatePostFixCSVs** postfix.
HH	Hour (2 digit), for example, 01, 12, 13 and so on.
H+	Hour (1 or 2 digits)
NN	Minute (2 digit)
N+	Minute (1 or 2 digits)
SS	Second (2-digit)
S+	Second (1 or 2 digits)
ZZZ	Time Zone (for example, GMT, EST, PST, and so on.)

For example:

ImportExtractDateToFormat0=D+/SHORTMONTH/YYYY

In this example dates are stored in the format **D+/SHORTMONTH/YYYY** (for example, 2/Jan/2002).

DateMonthCSVs

Allows you to specify the short months that the **ImportExtractDateFormatCSVs** and **ImportExtractDateToFormat** parameters' **SHORTMONTH** string uses.

For example:

DateMonthCSVs=Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec

In this example, the **SHORTMONTH** string uses short English months.

DateMonthCSVs=Jan,Fev,Mar,Avr,Mai,Juin,Juil,Aout,Sept,Oct,Nov,Dec

In this example, the **SHORTMONTH** string uses short French months.

DateLongMonthCSVs

Allows you to specify the long months that the **ImportExtractDateFormatCSVs** and **ImportExtractDateToFormat** parameters' **LONGMONTH** string uses.

For example:

DateLongMonthCSVs=January,February,March,April,May,June,July,August,September, October,November,December

In this example, the **LONGMONTH** string uses long English months.

DateLongMonthCSVs=Janvier,Février,Mars,Avril,Mai,Juin,Juillet,Août,Septembre,Octob re,Novembre,Décembre

In this example, the **LONGMONTH** string uses long French months.

DatePostfixCSVs

Allows you to specify the postfix that the **ImportExtractDateFormatCSVs** and **ImportExtractDateToFormat** parameters' **LONGDAY** string uses.

For example:

DatePostfixCSVs=1st,2nd,3rd,4th,5th

In this example, the **LONGDAY** string uses English postfixes.

DatePostfixCSVs=1.,2.,3.,4.,5.

In this example, the **LONGDAY** string uses German postfixes.

ImportExtractLength

Enter **1** if you want to extract the file length into the **ImportExtractLengthToField** IDOL server field.

Enter **0** if you do not want to extract the file length

ImportExtractLengthToField

If you have set **ImportExtractLength** to **1**, **ImportExtractLengthToField** allows you to specify the name of the IDOL server field into which the file length will be extracted.

For example:

ImportExtractLengthToField=FileLength

In this example the file length is stored in the **FileLength** IDOL server field.

ImportFieldGlueSourceCSVs*n*

Allows you to combine the values of multiple field in a new field. Specify the fields whose values you want to glue into the field value of the corresponding **ImportFieldGlueDestination*n*** field. You can also glue strings into the **ImportFieldGlueDestination*n*** field.

You can use DRECONTENT, DREFERENCE, DRETITLE, DREDBNAME or DREFILENAME as destination but not as source fields.

Note: you must separate the values and strings that you want to glue together with commas.

For example:

ImportFieldGlueSourceCSVs0=FnameFirstName, ,FnameSurName

ImportFieldGlueDestination0=FullName

In this example, the value of the **FirstName** field and the **SurName** field is glued into the **FullName** field. A space string is glued between the values of the name fields. If the **FirstName** field has the value **John** and the **SurName** field has the value **Smith**, the **FullName** field will contain the value **John Smith**.

ImportFieldGlueDestination*n*

Allows you to specify the name of the destination field into which you want to glue the fields specified in **ImportFieldGlueSourceCSVs*n***.

You can use DRECONTENT, DREFERENCE, DRETITLE, DREDBNAME or DREFILENAME as destination but not as source fields.

For example:

ImportFieldGlueSourceCSVs0=FnameName1, and ,FnameName2

ImportFieldGlueDestination0=Title

In this example, the value of the **Name1** field and the **Name2** field is glued into the **Title** field. The string **" and "** is glued between the values of the name fields. If the **Name1** field has the value **Romeo** and the **Name2** field has the value **Juliet**, the **Title** field will contain the value **Romeo and Juliet**.

ImportFieldHTMLConvertChars

Enter **true** if you want to replace HTML characters contained in fields with equivalent characters. By default this is **false**.

For example:

If a document contains the field **Name=John Smith** and you have set **ImportFieldHTMLConvertChars** to **true**, the **Name** field is imported with the value **John Smith**.

ImportMetaToFields

Enter **true** if you want to store the values of HTML meta tags in IDOL server fields (this is the default setting). You must specify these fields in the IDOL server configuration file before the documents are indexed.

For example:

If a file that you want to import contains the meta tag **<meta http-equiv="Germatic" content="efficiency of communication">** or **<meta name="Germatic" content="efficiency of communication">**, and you have set **ImportMetaToFields** to **true**, the document is imported with the IDOL server field **Germatic=efficiency of communication**.

ImportMinTitleChars

Allows you to specify the minimum amount of characters that can be used as a title for imported documents. By default this is **3**.

If a title of less than the specified number of characters is found, the title is taken from the document content instead.

ImportRemapField*n*

Allows you to remap fields. Specify the fields whose value you want to remap to the value of the corresponding **ImportRemapFieldTo*n*** fields. This useful, for example, if documents have different fields that contain the same data type, and you want to store this data in a uniform field.

For example:

ImportRemapField0=Logged

ImportRemapField1=Saved

ImportRemapFieldTo0=Date

ImportRemapFieldTo1=Date

In this example, the **Logged** field and the **Saved** field contain a date. When documents that contain **Logged** or **Saved** fields are imported, the date values that these fields contain are stored in a **Date** field.

ImportRemapFieldTo*n*

Allows you to remap fields. Specify the fields in which you want to store the values of the corresponding **ImportRemapField*n*** fields. This useful, for example, if documents have different fields that contain the same data type, and you want to store this data in a uniform field.

For example:

ImportRemapField0=Logged

ImportRemapField1=Saved

ImportRemapFieldTo0=Date

ImportRemapFieldTo1=Date

In this example, the **Logged** field and the **Saved** field contain a date. When documents that contain **Logged** or **Saved** fields are imported, the date values that these fields contain are stored in a **Date** field.

ImportFieldOpApplyTo*n*

Allows you to specify fields, to which you want to apply the corresponding **ImportFieldOp*n*** operations. Use **ImportFieldOpParam*n*** to specify appropriate parameters for the specified operation.

For example:

ImportFieldOp0=STRIPCHARS

ImportFieldOpApplyTo0=Author

ImportFieldOpParam0=&!#

In this example the characters **&**, **!** and **#** are removed from the **Author** field during the importing process.

ImportFieldOp*n*

Allows you to specify one or more operations that you want to apply to the corresponding **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify appropriate parameters for the specified operation.

For example:

ImportFieldOp0=STRIPCHARS

ImportFieldOpApplyTo0=Author

ImportFieldOpParam0=&!#

In this example the characters **&**, **!** and **#** are removed from the **Author** field during the importing process.

ImportFieldOpParam*n*

Allows you to specify the parameters that the corresponding **ImportFieldOp*n*** operation requires, which you are applying to the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOp0=REPLACESTRING

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOpParam0="£500";"£445.99"

In this example the first occurrence of the string **£500** in the **DRECONTENT** field is replaced with the string **£445.99** during the importing process.

ImportFieldOpCheckField*n*

Allows you to specify a field that must contain one of the strings you specify with the corresponding **ImportFieldOpCheckValue0**. If the **ImportFieldOpCheckField*n*** field doesn't contain one of the specified strings, the corresponding **ImportFieldOp*n*** is not performed.

For example:

ImportFieldOp0=REPLACESTRING

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOpParam0="£500";"£445.99"

ImportFieldOpCheckField0=DREFILENAME

ImportFieldOpCheckValue0=*.doc,*.txt

In this example, the **REPLACESTRING** field operation is only performed if the **DREFILENAME** field for a document contains a name with the extension **.doc** or **.txt**.

ImportFieldOpCheckValue*n*

Allows you to specify one or more strings that the field you specify with the corresponding **ImportFieldOpCheckField*n*** must contain. If the field doesn't contain one of the strings that you specify with **ImportFieldOpCheckValue*n***, the corresponding **ImportFieldOp*n*** is not performed.

Note: if you want to specify multiple strings, they must be separated by commas.

For example:

ImportFieldOp0=REPLACESTRING

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOpParam0="£500";"£445.99"

ImportFieldOpCheckField0=DREFILENAME

ImportFieldOpCheckValue0=*.doc,*.txt

In this example, the **REPLACESTRING** field operation is only performed if the **DREFILENAME** field for a document contains a name with the extension **.doc** or **.txt**.

ImportExtractExtension

Enter **true** if you want to extract the extension of the document that you are importing to the **ImportExtractExtensionToField** field.

Enter **false** if you don't want to store the document's extension in a field. This is default.

ImportExtractExtensionToField

If you have set **ImportExtractExtension** to **true**, you can specify the name of the field in which you want to store the file extension of the document that you are importing.

Field operations

Field operations allow you to modify fields before they are imported into IDOL server. You can specify the following field operations for the **ImportFieldOp*n*** parameter. Use the **ImportFieldOpApplyTo*n*** parameter to specify which fields you want to apply a field operation to, and the **ImportFieldOpParam*n*** parameter to specify any parameters that the field operation requires.

Note: the name that you give to the fields must not start with the string **DRE**. Field names that start with **DRE** are reserved for Autonomy (for example, **DRECONTENT**).

Base64Decode

If the text in the **ImportFieldOpApplyTo*n*** field is Base64 encoded, this operation will decode it.

BlankString

Allows you to replace a string in the **ImportFieldOpApplyTo*n*** field with a space.

For example:

ImportFieldOpApplyTo0=Date

ImportFieldOp0=BlankString

ImportFieldOpParam0=4th

In this example, if the **Date** field contains the value **January 4th**, the string **4th** is replaced with a space, and the **Content** field is imported with the value **January**.

Escape

Allows you to escape the text contained in the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOp0=Escape

ImportFieldOpApplyTo0=Name

In this example, if the **Name** field contains the value **John Smith**, the text is escaped, and the **Name** field is imported with the value **John%20Smith**.

Unescape

Unescapes escaped text that the **ImportFieldOpApplyTo*n*** field contains.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=Unescape

In this example, if the **Name** field contains the value **John%20Smith**, the text is unescaped and the **Name** field is imported with the value **John Smith**.

EncryptSecurityField

Allows you to encrypt the security string that the **ImportFieldOpApplyTo*n*** field contains.

For example:

ImportFieldOpApplyTo0=MySecurityField

ImportFieldOp0=EncryptSecurityField

StartAtChars

Allows you to import the content of the **ImportFieldOpApplyTo*n*** field from a specified character onwards. Enter one of the following for **ImportFieldOpParam*n*** to determine up to which character the field content is discarded.

<character>,0

Deletes the content of the field up to the first instance of the specified **<character>**.

<character>,1

Deletes the content of the field up to the last instance of the specified **<character>**.

<character>,2

Deletes the first instance of the specified **<character>** from the field and the content up to this character.

<character>,3

Deletes the last instance of the specified **<character>** from the field and the content up to this character.

For example:

ImportFieldOpApplyTo0=Vitamin

ImportFieldOp0=StartAtChars

In this example, if the **Vitamin** field contains the string **A B1 B2**, part of this string is deleted as follows, depending on **ImportFieldOpParam0**:

ImportFieldOpParam0="B","0"

The **Vitamin** field is imported with the value **B1 B2**.

ImportFieldOpParam0="B","1"

The **Vitamin** field is imported with the value **B2**.

ImportFieldOpParam0="B","2"

The **Vitamin** field is imported with the value **1 B2**.

ImportFieldOpParam0="B","3"

The **Vitamin** field is imported with the value **2**.

EndAtChars

Allows you to import the content of the **ImportFieldOpApplyTo*n*** field up to a specified character. Enter one of the following for **ImportFieldOpParam*n*** to determine from which character on the field content is discarded.

<character>,0

Deletes the content of the field after the first instance of the specified **<character>**.

<character>,1

Deletes the content of the field after the last instance of the specified **<character>**.

<character>,2

Deletes the first instance of the specified **<character>** from the field and the content that follows this character.

<character>,3

Deletes the last instance of the specified **<character>** from the field and the content that follows this character.

For example:

ImportFieldOpApplyTo0=Animal

ImportFieldOp0=EndAtChars

In this example, if the **Animal** field contains the string **Hippopotamus**, part of this string is deleted as follows, depending on **ImportFieldOpParam0**:

ImportFieldOpParam0="P","0"

The **Animal** field is imported with the value **Hip**.

ImportFieldOpParam0=P","1"

The **Animal** field is imported with the value **Hippop**.

ImportFieldOpParam0="P","2"

The **Animal** field is imported with the value **Hi**.

ImportFieldOpParam0="P","3"

The **Animal** field is imported with the value **Hippo**.

GetBetween

Allows you to import strings from between two specified delimiter strings in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify the start and end delimiters you want to find (in the format **<start_string>,<end_string>**). **GetBetween** reads the strings between the specified delimiters and the **ImportFieldOpApplyTo*n*** field is imported with these values in a comma separated list.

For example:

ImportFieldOpApplyTo0=Numbers

ImportFieldOp0=GetBetween

ImportFieldOpParam0=<num>,</num>

In this example, if the **Numbers** field contains the value **Numbers: <num>one</num> and <num>two</num> and <num>three</num>**, it is imported with the value **one,two,three**.

EliminateBetweenStrings

Allows you to remove a string between two specified strings in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify the start and end delimiters you want to find (in the format **$<start_string>$<end_string>**). **EliminateBetweenStrings** removes the strings between the specified delimiters and the **ImportFieldOpApplyTo*n*** field is imported with the remaining values.

For example:

ImportFieldOpApplyTo0=MyString

ImportFieldOp0=EliminateBetweenStrings

ImportFieldOpParam0=$first! $third!

In this example, if the **MyString** field contains the value **first! second! third! fourth!** , it is imported with the value **first! third! fourth!** .

EliminateBetweenStringsInclusive

Allows you to remove a string that starts and ends with two strings that you specify in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify the start and end delimiters you want to find (in the format **$<start_string>$<end_string>**). **EliminateBetweenStringsInclusive** removes the entire string from the start delimiter to the end delimiter, including the delimiters themselves, and the **ImportFieldOpApplyTo*n*** field is imported with the remaining values.

For example:

ImportFieldOpApplyTo0=MyString

ImportFieldOp0=EliminateBetweenStringsInclusive

ImportFieldOpParam0=$first! $third!

In this example, if the **MyString** field contains the value **first! second! third! fourth!** , it is imported with the value **fourth!** .

EliminateBetweenChars

Allows you to remove a string between two specified characters in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** as follows to specify how this string is removed:

ImportFieldOpParam*n* = <start_char>,<end_char>,<whether_to_remove_start_char>,<whether_to_remove_end_char>

<start_char>

Enter the character that marks the start of the string you want to remove.

<end_char>

Enter the character that marks the end of the string you want to remove.

<whether_to_remove_start_char>

Enter **0** if you want to include the character you specified with **<start_char>** in the string that is removed.

Enter **1** if you don't want to include the character you specified with **<start_char>** in the string that is removed.

<whether_to_remove_end_char>

Enter **0** if you want to include the character you specified with **<end_char>** in the string that is removed.

Enter **1** if you don't want to include the character you specified with **<end_char>** in the string that is removed.

For example:

ImportFieldOpApplyTo0=Alphabet

ImportFieldOp0=EliminateBetweenChars

ImportFieldOpParam0=a,z,0,1

In this example, if the **Alphabet** field contains the value **a,b … y,z**, it is imported with the value **a**.

TruncateAfterOccurrences

Allows you to discard any content from the **ImportFieldOpApplyTo*n*** field that occurs after the **n**th instance of a specified string. Use **ImportFieldOpParam*n*** to specify the string after which you want content to be discarded and the number of times that this string is permitted to occur in the field before content is discarded. Note that **ImportFieldOpParam*n*** takes the following format:

<string>:<number>

<string>

The string after which content is discarded.

<number>

The number of times that the **<string>**is permitted to occur in the **ImportFieldOpApplyTo*n*** field before content is discarded.

For example:

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOp0=TruncateAfterOccurrences

ImportFieldOpParam0=end:5

In this example, the Import module identifies the **5th** occurrence of the string **end** and discards any content that the **DRECONTENT** field contains from this point onwards.

ImportFile

Allows you to import the files whose names are listed in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify the path to the location where the listed files are stored.

For example:

ImportFieldOpApplyTo0=MyField

ImportFieldOp0=ImportFile

ImportFieldOpParam0=C:\Documents\Files_for_importing

In this example, the Import module reads the name of the files that it should import from the **MyField** field, and imports each one of them from **C:\Documents\Files_for_importing** .

Note:

If some of the listed files contain fields that have the same name, only the first instance of these fields is imported. Fields that are subsequently imported overwrite any previously imported field of the same name. This also applies to the files' Content fields.

You can specify whether you want Content fields to be overwritten and whether you want the original file to be deleted when it has been imported, by adding one of the following values before the directory path you specify with **ImportFieldOpParam*n***:

0; Overwrite the value of the first Content field with subsequently imported Content fields.
Don't delete the file you are importing from when importing is complete.

1; Don't overwrite Content field values with subsequently imported Content fields. (Content field values are appended to the existing Content field value instead).
Don’t delete the file you are importing from when importing is complete.

2; Overwrite the value of the first Content field with subsequently imported Content fields.
Delete the file you are importing from when importing is complete.

3; Don't overwrite Content field values with subsequently imported Content fields. (Content field values are appended to the existing Content field value instead).
Delete the file you are importing from when importing is complete.

For example:

ImportFieldOpApplyTo0=MyField

ImportFieldOp0=ImportFile

ImportFieldOpParam0=3;C:\Documents\Files_for_importing

In this example, the Import module reads the name of the files that it should import from the **MyField** field, and imports each one of them from **C:\Documents\Files_for_importing** . The first instance of each field that the files contain is overwritten by fields of the same name that are subsequently imported. An exception to this is the Content field. The first instance of this field is imported and the contents of subsequently imported Content fields are added to the contents of the first imported Content field. When the source file has been imported, it is deleted.

ImportURL

Allows you to import the pages whose URLs are listed in the **ImportFieldOpApplyTo*n*** field. Use **ImportFieldOpParam*n*** to specify the path to the location where the listed pages are stored.

For example:

ImportFieldOpApplyTo0=MyField

ImportFieldOp0=ImportURL

ImportFieldOpParam0=http://www.mycompany.com/intranet_location/

In this example, the Import module reads the name of the pages that it should import from the **MyField** field, and imports each one of them from **http://www.mycompany.com/intranet_location/**.

Notes:

- If you connect to the internet via a proxy server, you must specify the settings for your proxy server with **ImportURLProxyHost**, **ImportURLProxyServer**, **ImportURLProxyUsername** and **ImportURLProxyPassword**. Please refer to the **Importing web pages** section of this chapter for details of these settings.

- If some of the listed pages contain fields that have the same name, only the first instance of these fields is imported. Fields that are subsequently imported overwrite any previously imported field of the same name. This also applies to the pages' Content fields.

  If you do not want the value of the first Content field to be overwritten by subsequently imported Content fields, but want to append the contents of subsequently imported Content fields to the contents of the first one, you can prefix the **ImportFieldOpParam*n*** string with **1;**

  For example:

  ImportFieldOpApplyTo0=MyField

  ImportFieldOp0=ImportURL

  ImportFieldOpParam0=1; http://www.mycompany.com/intranet_location/

  In this example, the Import module reads the name of the pages that it should import from the **MyField** field, and imports each one of them from **http://www.mycompany.com/intranet_location/**. The first instance of each field that the pages contain is overwritten by fields of the same name that are subsequently imported. An exception to this is the Content field. The first instance of this field is imported and the contents of subsequently imported Content fields are added to the contents of the first imported Content field.

StartAtCharsCSVs

If the **ImportFieldOpApplyTo*n*** field contains a comma-separated list of strings (for example, file names), **StartAtCharsCSVs** allows you to import each listed string from a specified character onwards. Enter one of the following for **ImportFieldOpParam*n*** to determine up to which character the individual string is discarded:

<character>,0

Deletes the strings listed in the **ImportFieldOpApplyTo*n*** field up to the first instance of the specified **<character>**.

<character>,1

Deletes the strings listed in the **ImportFieldOpApplyTo*n*** field up to the last instance of the specified **<character>**.

<character>,2

Deletes the first instance of the specified **<character>** from each string listed in the **ImportFieldOpApplyTo*n*** field, and each string up to this character.

<character>,3

Deletes the last instance of the specified **<character>** from each string listed in the **ImportFieldOpApplyTo*n*** field, and each string up to this character.

For example:

ImportFieldOpApplyTo0=MyField

ImportFieldOp0=StartAtCharsCSVs

In this example, if the **MyField** field contains the string **temp/test/files,mydocuments/archive, current/save/QA**, part of this string is deleted as follows, depending on **ImportFieldOpParam0**:

ImportFieldOpParam0="/","0"

The **MyField** field is imported with the value **/test/files,/archive,/save/QA**.

ImportFieldOpParam0="/","1"

The **MyField** field is imported with the value **/files,/archive,/QA**.

ImportFieldOpParam0="/","2"

The **MyField** field is imported with the value **test/files,archive,save/QA**.

ImportFieldOpParam0="/","3"

The **MyField** field is imported with the value **files,archive,QA**.

ExtractDigits

Allows you to extract digits from the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Price

ImportFieldOp0=ExtractDigits

In this example, if the **Price** field contains the value **£24**, the digits are extracted and the field is imported with the value **24**.

ExtractPrice

Allows you to extract **£** or **$** symbols.

For example:

ImportFieldOpApplyTo0=Price

ImportFieldOp0=ExtractPrice

In this example, if the **Price** field contains the value **£24**, the **£** symbol is extracted and the field is imported with the value **£**.

Note that if you want to import the currency symbol with the price, you must specify **ExtractPrice** and **ExtractDigits**.

For example:

ImportFieldOpApplyTo0=Price

ImportFieldOp0=ExtractDigits

ImportFieldOp0=ExtractPrice

In this example, if the **Price** field contains the value **£24**, the **£** symbol is extracted and the digits are extracted. The Price field is then imported with the value **£24**.

GobbleChars

Allows you to specify one or more characters, in order to reduce adjacent instances of these characters to one instance in the **ImportFieldOpApplyTo*n*** field.

Note: you must not separate the characters that you specify with commas

For example:

ImportFieldOpApplyTo0=DREtitle

ImportFieldOp0=GOBBLECHARS

ImportFieldOpParam0=LG

In this example, if the **DREtitle** field has the value **HALLOGGEN**, all adjacent multiple instances of the letter **L** are reduced to one **L**, and all adjacent multiple instances of the letter **G** are reduced to one **G**, so that the **DREtitle** field is imported with the value **HALOGEN**.

GobbleWhiteSpace

Allows you to reduce multiple adjacent spaces in the **ImportFieldOpApplyTo*n*** field to one space.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=GobbleWhiteSpace

In this example, if the **Name** field contains the value **John Smith**, the spaces between the first name and the surname are reduced to one space, so that the **Name** field is imported with the value **John Smith**.

ReplaceLineWithSpace

Allows you to replace any carriage returns in the **ImportFieldOpApplyTo*n*** field with a space.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=ReplaceLineWithSpace

In this example, if the **Name** field has the value **John¶Smith**, the carriage return between the first name and the surname is replaced with a space, and the **Name** field is imported with the value **John Smith**.

ReplaceString

Allows you to replace the first occurrence of a string in the **ImportFieldOpApplyTo*n*** field with another string .

For example:

ImportFieldOpApplyTo0=DREContent

ImportFieldOp0=ReplaceString

ImportFieldOpParam0="£500";"£445.99"

In this example the first occurrence of the string **£500** in the **DREContent** field is replaced with the string **£445.99** during the importing process.

ReplaceMultiple

Allows you to replace all occurrences of a specific string in the **ImportFieldOpApplyTo*n*** field with another string .

For example:

ImportFieldOpApplyTo0=DREReference

ImportFieldOp0=ReplaceMultiple

ImportFieldOpParam0=ü;%fc

In this example any occurrence of the string **ü** in the **DREContent** field is escaped during the importing process.

StringMatch

Allows you to replace the **ImportFieldOpApplyTo*n*** field value with one of two specified strings. Use **ImportFieldOpParam*n*** as follows to determine with which of the two strings the **ImportFieldOpApplyTo*n*** field value is replaced:

> **ImportFieldOpParam*n*=<ImportFieldOpApplyTo*n*_value>,<match_replacement>, <no_match_replacement>**
>
> **<ImportFieldOpApplyTo*n*_value>**
>
> Enter the string that the **ImportFieldOpApplyTo*n*** field must contain to be replaced with the **<match_replacement>** string. If the **ImportFieldOpApplyTo*n*** field does not contain the specified string, it's value is replaced by the **<no_match_replacement>** string.
>
> **<match_replacement>**
>
> Enter the string with which you want to replace the **ImportFieldOpApplyTo*n*** field value, if it matches the specified **<ImportFieldOpApplyTo*n*_value>**.
>
> **<no_match_replacement>**
>
> Enter the string with which you want to replace the **ImportFieldOpApplyTo*n*** field value, if it does not match the specified **<ImportFieldOpApplyTo*n*_value>**.

For example:

> **ImportFieldOpApplyTo0=Animal**
>
> **ImportFieldOp0=StringMatch**
>
> **ImportFieldOpParam0="elephant";"jumbo";"hippo"**

In this example, if the **Animal** field value is **elephant**, the field is imported with the value **jumbo**. If the **Animal** field value is not **elephant**, it is imported with the value **hippo**.

Alternatively, you can use **Fname** to specify field names for your parameters, for example:

> **ImportFieldOpApplyTo0=Animal**
>
> **ImportFieldOp0=StringMatch**
>
> **ImportFieldOpParam0=FnameMyFavouriteAnimal;"jumbo";FnameAnotherAnimal**

In this example, if the **Animal** field value matches the **MyFavouriteAnimal** field value, the **Animal** field is imported with the value **jumbo**. If the **Animal** field value does not match the **MyFavouriteAnimal** field value, the **Animal** field is imported with the **AnotherAnimal** field value.

StartFromLetter

Allows you to discard any numbers that precede text in the **ImportFieldOpApplyTo*n*** field and import the field content from the first letter onwards.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=StartFromLetter

In this example, if the **Name** field has the value **2John Smith**, the **2** before the name is discarded and the **Name** field is imported with the value **John Smith**.

ToLower

Allows you to replace all upper case letters with lower case letters in the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=ToLower

In this example, if the **Name** field has the value **John Smith**, the upper case letters are converted to lower case and the **Name** field is imported with the value **john smith**.

ToUpper

Allows you to replace all lower case letters with upper case letters in the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=ToUpper

In this example, if the **Name** field has the value **John Smith**, the lower case letters are converted to upper case and the **Name** field is imported with the value **JOHN SMITH**.

StripChars

Allows you to remove one or more specified characters from the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=StripChars

ImportFieldOpParam0=”!#”

In this example, if the **Name** field has the value **#John Smith!**, the characters **#** and **!** are removed, and the **Name** field is imported with the value **John Smith**.

StripHTML

Allows you to remove any remaining HTML tags from the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=StripHTML

In this example, if the **Name** field has the value **Name=John Smith**, the italic HTML tags (**** and ****) are removed, and the **Name** field is imported with the value **John Smith**.

If you want to convert any HTML entities that the **ImportFieldOpApplyTo*n*** field contains to equivalent strings, you must set **ImportFieldOpParamN** to **HTMLConvertEntities**.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=StripHTML

ImportFieldOp0=HTMLConvertEntities

In this example, if the **Name** field has the value **John Smith**, the HTML entities are converted and the italic HTML tags are removed, and the **Name** field is imported with the value **John Smith**.

StripNumbers

Allows you to discard any numbers that the **ImportFieldOpApplyTo*n*** field contains.

For example:

ImportField0pApplyTo0=Street

ImportFieldOp0=StripNumbers

In this example, if the **Street** field has the value **4 Baker Street**, the **4** is removed during the importing process and the **Street** field is imported with the value **Baker Street**.

TailWhiteSpace

Allows you to discard any extra spaces from the end of the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=TailWhiteSpace

In this example, if the **Name** field has the value "**John Smith** ", the empty spaces that follow the name are discarded and the **Name** field is imported with the value "**John Smith**".

TopNTailWhiteSpace

Allows you to discard any extra spaces from the beginning and the end of the **ImportFieldOpApplyTo*n*** field.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=TopNTailWhiteSpace

In this example, if the **Name** field has the value " **John Smith** ", the empty spaces that precede and follow the name are discarded and the **Name** field is imported with the value "**John Smith**".

TerminateAtInvalid

Allows you to import the **ImportFieldOpApplyTo*n*** field up to an invalid character (that is, any character that has an ASCII value of 31 or less).

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=TerminateAtInvalid

In this example, if the **Name** field has the value **Name=John → Smith**, the tab (→) and any content that follows it are discarded, and the **Name** field is imported with the value **John**.

TerminateAtLine

Allows you to import the **ImportFieldOpApplyTo*n*** field up to a carriage return.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=TerminateAtLine

In this example, if the **Name** field has the value **Name=John ¶Smith**, the line break and any content that follows it are discarded and the **Name** field is imported with the value **John**.

TerminateAtSpace

Allows you to import the **ImportFieldOpApplyTo*n*** field up to a space.

For example:

ImportFieldOpApplyTo0=Name

ImportFieldOp0=TerminateAtSpace

In this example, if the **Name** field has the value **Name=John Smith**, any content after the space is discarded and the **Name** field is imported with the value **John**.

ConvertDate

Allows you to convert a date in the **ImportFieldOpApplyTo*n*** field from one format to another.

For example:

ImportFieldOpApplyTo0=Date

ImportFieldOp0=ConvertDate

ImportFieldOpParam0=YYYY/MM/DD;MM/DD/YY

In this example, the format of the **Date** field is converted from the format year/month/day to the format month/day/short year. That is, if the value of the **Date** field is **2002/11/01**, it is imported as **11/01/02**.

CharSetConvert

Allows you to convert the text in the **ImportFieldOpApplyTo*n*** field from one character set to another. Use **ImportFieldOpParam*n*** to specify the character sets to convert from and to in the following form:

ImportFieldOpParam*n*=<convert_from>:<convert_to>

<convert_from>

Enter the name of the current character set for the field.

<convert_to>

Enter the name of the character set to convert to.

For example:

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOp0=CharSetConvert

ImportFieldOpParam0=UTF8:HEBREW

In this example, the text in the **DRECONTENT** field is converted from **UTF8** to **HEBREW**.

ReverseTextDirection

Languages whose script is written from right to left (for example, Arabic, Hebrew, and so on) are implemented in one of the following ways:

- data is stored left to right but displayed by the application (for example, a browser) in right to left form.
- data is stored right to left

ReverseTextDirection allows you to reverse content data that is stored left to right, so that it is stored in right to left form.

For example:

ImportFieldOpApplyTo0=DRECONTENT

ImportFieldOp0=ReverseTextDirection

In this example, the text direction in the **DRECONTENT** field is reversed.

Note: you must ensure that the data has been written from right to left (for example, by using the **View Source** option when you display the data in a browser). If you set **ReverseTextDirection** for data that has been written from left to right and is only displayed as from right to left (for example, in a browser), it will be imported in reverse and become unreadable.

Selecting content for importing

The following settings allow you to specify how document data should be extracted to form the main content for indexing into IDOL server.

ImportBreaking

Enter **true** if you want to split large documents into smaller documents before they are indexed (recommended). This is the default.

If you enter **false** large documents are not split up into smaller documents.

Note: if you enable **ImportBreaking** and you are using a DRE version 3, you should set **Combine** to **1** in the configuration file of the DRE into which you are indexing the documents.

ImportBreakingMaxParagraphWords

If you have enabled **ImportBreaking**, **ImportBreakingMaxParagraphWords** allows you to specify the maximum size of the sections that a document is broken into. If a document is larger than **ImportBreakingMaxParagraphWords**, the Import module splits the document at the first sentence end after **ImportBreakingMinParagraphWords**, or at **ImportBreakingMaxParagraphWords** if no sentence end is found earlier.

By default, the value of this setting is **360**.

For example:

ImportBreaking=true

ImportBreakingMinDocWords=500

ImportBreakingMinParagraphWords=400

ImportBreakingMaxParagraphWords=600

In this example, **ImportBreaking** is enabled, so the Import module breaks all documents that contain more than **600** words into sections. A new section starts at the end of the first sentence after **400** words, or at exactly **600** words if there is no sentence end earlier.

ImportBreakingMinParagraphWords

If you have enabled **ImportBreaking**, **ImportBreakingMinParagraphWords** allows you to specify where you want section breaks to occur. The connector splits documents at the end of the first sentence after the number of words you specify with **ImportBreakingMinParagraphWords**, or at **ImportBreakingMaxParagraphWords** if no sentence end is found earlier.

By default the value of this setting is **160**.

For example:

ImportBreaking=true

ImportBreakingMinDocWords=500

ImportBreakingMinParagraphWords=400

ImportBreakingMaxParagraphWords=600

In this example, **ImportBreaking** is enabled, so the Import module breaks all documents that contain more than **600** words into sections. A new section starts at the end of the first sentence after **400** words, or at exactly **600** words if there is no sentence end earlier.

ImportBreakingMinDocWords

Enter a number of words that a document must contain to be split up into smaller documents (provided **ImportBreaking** is enabled). Any document that contains less than the number of words you specify is imported whole.

The default value for this setting is **360**.

Note:

- The connector attempts to split up the document at the end of a sentence within the range you specify with **ImportBreakingMinParagraphWords** and **ImportBreakingMaxParagraphWords**.
- **ImportBreakingMinDocWords** must be greater than **ImportBreakingMinParagraphWords**.

ImportStartDefCSVs

Enter one or more strings that mark the beginning of content in a document. Any text that the document contains between the specified strings and the corresponding **ImportEndDefCSVs** strings is imported into IDOL server.

If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportStartDefCSVs=Title,Name

ImportEndDefCSVs=Appendix,Glossary

In this example, any text that the document contains between **Title** or **Name** and **Appendix** or **Glossary** is imported into IDOL server.

ImportEndDefCSVs

Enter one or more strings that mark the end of content in a document. Any text that the document contains between the specified strings and the corresponding **ImportStartDefCSVs** strings is imported into IDOL server.

If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportStartDefCSVs=Title,Name

ImportEndDefCSVs=Appendix,Glossary

In this example, any text that the document contains between **Title** or **Name** and **Appendix** or **Glossary** is imported into IDOL server.

ImportPageBreakDefs

Allows you to specify a string that is used to mark a document break. This is used to split up documents into segments, so their content can be imported more easily into IDOL server. Each segment is contained in one IDX file. Once the IDX files have been indexed into IDOL server, the individual segments are joined back together to produce the original document.

For example:

Product 1
Data
Price
Product 2
Data
Price

If a document contains the above text, and you have set **ImportPageBreakDefs** to **Product**, the document is split into one segment for **Product 1** and another segment for **Product 2**.

ImportStartSkipWords

Allows you to specify the number of words to skip from the beginning of a document when importing a document.

For example:

ImportStartSkipWords=7

In this example, if the document that you are importing contains the text "**I do not want this text but I do want this text**", the first **7** words are discarded and “**I do want this text**” is imported.

ImportTitleStartSkipWords

Enter the number of words that you want to skip before obtaining a title from the document.

ImportStripLinks

Enter **true** if you want to strip any hypertext links from documents that are imported.

Enter **false** if you want to import documents with any hypertext links that they may contain.

ImportStripStringCSVs

Allows you to specify one or more strings that you want to be stripped from documents, before they are imported.

For example:

ImportStripStringCSVs=unwanted,unnecessary

In this example, any instances of the strings **unwanted** and **unnecessary** are stripped from a document.

ImportStripTagCSVs

Allows you to specify one or more tags for which you want content between start and end tags to be stripped. If you want to specify multiple tags, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportStripTagCSVs=h1,p

In this example content between **<h1>** and **</h1>**, as well as between ** ** and ** ** will be stripped.

ImportReverseTextDirection

Languages whose script is written from right to left (for example, Arabic, Hebrew, and so on) are implemented in one of the following ways:

- data is stored left to right but displayed by the application (for example a browser) in right to left form.
- data is stored right to left

Enter **true** if you want to import content data that is stored left to right. The import module will reverse the text so that it is stored in right to left form. You can use **ImportReverseTextDirectionExtns** to specify the file extensions for files in which you want to reverse the text direction.

Note: you must ensure that the data has been written from right to left (for example by using the **View Source** option when you display the data in a browser). If you set **ImportReverseTextDirection** to **true** for data that has been written from left to right and is only displayed as from right to left (for example in a browser), it will be imported in reverse and become unreadable.

Enter **false** if you don't want to reverse text. This is the default setting.

ImportReverseTextDirectionExtns

If you have set **ImportReverseTextDirection** to **true**, **ImportReverseTextDirectionExtns** allows you to specify one or more file extensions for the files for which you want to reverse the direction of the text. Multiple file extensions must be separated with commas (with no space before or after a comma).

For example:

ImportReverseTextDirectionExtns=*.doc,*.txt

In this example, the text direction is reversed in files that have the extension **.doc** or **.txt**.

The default value for this setting is *.*, that is, the direction of the text is reversed for all files.

Modifying IDOL server references

During the importing process, the Import module sets a reference for each document that you are importing. The following settings allow you to modify this reference before it is imported into IDOL server. This is useful to set references so that they can be launched easily from a web server, third party client application, bespoke application and so on.

Note: IDOL server references must always be unique.

ImportAddHTMLAnchornames

Enter **true** if you want to add the name of the anchor to the end of the title in the document reference for each title in the HTML page. This is the default.

Enter **false** if you don't want to add the name of the anchor.

ImportPathReplaceString

Enter the string with which you want to replace a document's reference up to the **ImportPathReplaceUpToSlash** slash.

For example:

ImportPathReplaceString=http://x/y/

ImportPathReplaceUpToSlash=3

In this example, if a document has the reference **C:\a\b\page**, the part of the reference that precedes the **3**rd slash is replaced with the **ImportPathReplaceString**, and the document reference is imported as **http://x/y/page**.

ImportPathReplaceUpToSlash

Enter a number to specify up to which slash you want to replace a document's reference with the **ImportPathReplacestring** string.

For example:

ImportPathReplaceUpToSlash=3

ImportPathReplaceString=http://x/y/

In this example, if a document has the reference **C:\a\b\page**, the part of the reference that precedes the **3**rd slash is replaced with the **ImportPathReplaceString**, and the document reference is imported as **http://x/y/page**.

ImportRefReplaceCSVs

Allows you to specify which parts of a document's reference you want to replace. Enter one or more strings that you want to be replaced with **ImportRefReplaceWithCSVs**.

If you specify multiple strings, you must separate the individual strings with a comma (there must not be any space before or after the comma).

For example:

ImportRefReplaceCSVs=c:\inetpub\wwwroot,

ImportRefReplaceWithCSVs=http://127.0.0.1/,/

In this example every instance of the string **c:\inetpub\wwwroot** in the document's reference is replaced with **http://127.0.0.1/** and every instance of the string **** is replaced with **/**. (That is if the document's reference is **c:\inetpub\wwwroot\products\public**, it is imported as **http://127.0.0.1/products/public**).

ImportRefReplaceWithCSVs

Enter one or more strings with which you want to replace the strings that you have specified for **ImportRefReplaceCSVs**.

For example:

ImportRefReplaceCSVs=c:\inetpub\wwwroot,

ImportRefReplaceWithCSVs=http://127.0.0.1/,/

In this example every instance of the string **c:\inetpub\wwwroot** in the document's reference is replaced with **http://127.0.0.1/** and every instance of the string **** is replaced with **/**. (That is if the document's reference is **c:\inetpub\wwwroot\products\public**, it is imported as **http://127.0.0.1/products/public**).

Note:

You can use the following wildcards for **ImportRefReplaceCSVs** if you want to add the **ImportRefReplaceCSVs** strings to a document's reference instead of replacing it:

^ adds a string to the beginning of the document's reference.

$ adds a string to the end of the document's reference.

For example:

ImportRefReplaceCSVs=^,$

ImportRefReplaceWithCSVs=http://www.,.com

In this example the string **http://www.** is added to the beginning of the document's reference, and the string **.com** is added to the end of the document's reference. (That is if the document's reference is **autonomy**, it is imported as **http://www.autnomy.com**).

ImportRefTruncateAfter

Enter a number to specify how many times the **ImportRefTruncateString** can be contained in a document reference before the reference is truncated.

For example:

ImportRefTruncateAfter=2

ImportRefTruncateString=ab

In this example, if a document's reference is **C:\ab\cd\ab\ef\ab\page**, it is truncated after the **2**nd occurrence of the string **ab**, and imported as **C:\ab\cd\ab**.

ImportRefTruncateString

Allows you to specify a string after which the document reference is truncated. Use **ImportRefTruncateAfter** to specify after how many occurrences of this string the reference is truncated.

For example:

ImportRefTruncateString=ab

ImportRefTruncateAfter=2

In this example, if a document's reference is **C:\ab\cd\ab\ef\ab\page**, it is truncated after the **2**nd occurrence of the string **ab**, and imported as **C:\ab\cd\ab**.

Importing HTML files

The following settings allow you to determine which parts of HTML files are imported and to manipulate HTML content.

ImportHTMLExtns

Allows you to specify one or more file extensions that should be treated as HTML when they are imported.

If you want to specify multiple extensions, you need to separate them with commas (there should be no space before or after the comma).

For example:

ImportHTMLExtns=*.htm,*.shtml,*.sgml

In this example, if a file with the extension **.htm**, **.shtml** or **.sgml** is retrieved, it will be imported as an HTML file.

ImportHTMLConvertChars

Enter **true** if you want to replace all HTML entities that a document contains with equivalent characters before the document is imported. This is default.

Enter **false** if you want to import documents with their HTML entities.

HTMLImportStartDefFlags

Enter one of the following to indicate from where in the HTML the Import module should start to import:

0 The header and body of the HTML file are imported.

1 The header is ignored and only the body is imported. This is the default.

HTMLImportStartDefCSVs

Enter one or more strings that mark the beginning of content in an HTML document. Any text that the document contains between the specified strings and the corresponding **HTMLImportEndDefCSVs** strings is imported into IDOL server.

Note: if you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

**HTMLImportStartDefCSVs=<table>, **

**HTMLImportEndDefCSVs=</table>, **

In this example, any text that the document contains between **<table>** or ** ** and ** ** or **</table>** is imported into IDOL server.

HTMLImportEndDefCSVs

Enter one or more strings that mark the end of content in an HTML document. Any text that the document contains between the specified strings and the corresponding **HTMLImportStartDefCSVs** strings is imported into IDOL server.

Note: if you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

**HTMLImportStartDefCSVs=<table>, **

**HTMLImportEndDefCSVs=</table>, **

In this example, any text that the document contains between **<table>** or ** ** and **</table>** or ** ** is imported into IDOL server.

HTMLImportTurnToSpaceStartDefCSVs

If you have set **HTMLImportTurnToSpaceStripTags** to **true**, you can specify one or more strings for **HTMLImportTurnToSpaceStartDefCSVs** and **HTMLImportTurnToSpaceEndDefCSVs**. When an HTML document is imported the content between these strings and the strings themselves are removed.

If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

```
<html>
<head>
</head>
<body>
    Wanted Content
        <table> START
            Unwanted Content
        END </table>
    Wanted Content
</body>
</html>
```

If an HTML document contains the above and you have specified the following settings, the **Unwanted Content** and the **<table> START** and **END </table>** strings are removed when the document is imported.

HTMLImportTurnToSpaceStripTags=true

HTMLImportTurnToSpaceStartDefCSVs =<table> START

HTMLImportTurnToSpaceEndDefCSVs=END </table>

HTMLImportTurnToSpaceEndDefCSVs

If you have set **HTMLImportTurnToSpaceStripTags** to **true**, you can specify one or more strings for **HTMLImportTurnToSpaceStartDefCSVs** and **HTMLImportTurnToSpaceEndDefCSVs**. When an HTML document is imported the content between these strings and the strings themselves are removed.

If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

```
<html>
<head>
</head>
<body>
    Wanted Content
        <table> START
            Unwanted Content
        END </table>
    Wanted Content
</body>
</html>
```

If an HTML document contains the above and you have specified the following settings, the **Unwanted Content** and the **<table> START** and **END </table>** strings are removed when the document is imported.

HTMLImportTurnToSpaceStripTags=true

HTMLImportTurnToSpaceStartDefCSVs =<table> START

HTMLImportTurnToSpaceEndDefCSVs=END </table>

HTMLImportTurnToSpaceStripTags

Enter **true** if you want to remove content between the **HTMLImportTurnToSpaceStartDefCSVs** and **HTMLImportTurnToSpaceEndDefCSVs** strings, and the strings themselves from HTML documents.

Enter **false** if you do not want to remove the content between the **HTMLImportTurnToSpaceStartDefCSVs** and **HTMLImportTurnToSpaceEndDefCSVs** strings, and the strings themselves.

Converting non-HTML/text documents to HTML

When you import non-HTML/Text documents, a Slave program converts these documents into HTML format. The settings in this section determine the file types for which the rendered files are saved and how the reference to these copies are modified before they are indexed into IDOL server.

Keeping HTML rendered files is useful for providing HTML alternatives to existing documents. This generally speeds up access from a browser as HTML files are quicker to download than most binary formats available.

ImportRenderedHTMLExtensions

Allows you to specify the extensions of files that you want to convert to HTML when they are imported.

If you want to specify multiple extensions, you must separate them with commas (there must be no space before or after a comma).

For example:

ImportRenderedHTMLExtensions=*.doc,*.xls

In this example, all files that have a **doc** or **xls** extension are converted to HTML when they are imported.

ImportRenderedHTMLFieldName

Allows you to specify the name of the field that will contain the fully qualified path (which is made up of the **ImportRenderedHTMLMoveToDir** path and the name of the rendered HTML file) to the rendered HTML file that is produced by a Slave when you are importing a non-HTML file.

For example:

ImportRenderedHTMLFieldname=RenderedHTMLCopy

ImportRenderedHTMLMoveToDir=C:\Directory\Subdirectory\Copies

In this example, if the rendered HTML file has the name **Document.htm**, the **RenderedHTMLCopy** field is imported with the value **C:\Directory\Subdirectory\Copies\Document.htm**.

ImportRenderedHTMLMoveToDir

Enter the path to the directory where you want to store the rendered HTML files that are generated by the Slaves for non-HTML documents.

ImportRenderedHTMLPathReplaceString

Specify the string that you want to replace the string preceding the **ImportRenderedHTMLPathReplaceUpToSlash** slash in a reference to a rendered HTML file (stored in the **ImportRenderedHTMLFieldName** field).

For example:

ImportRenderedHTMLPathReplacestring=http://x/y/

ImportRenderedHTMLPathReplaceUpToSlash=3

In this example, if the reference to the rendered HTML file is **C:\a\b\document.htm**, the string before the 3rd slash is replaced with http://x/y/ and the document is imported with the document reference **http://x/y/document.htm**.

ImportRenderedHTMLPathReplaceUpToSlash

Enter a number to specify which slash in a rendered HTML file's reference (stored in the **ImportRenderedHTMLFieldName** field) marks the end of the string that you want to replace with the string specified for **ImportRenderedHTMLPathReplaceString**.

For example:

ImportRenderedHTMLPathReplaceString=http://x/y/

ImportRenderedHTMLPathReplaceUpToSlash=3

In this example, if the reference to the rendered HTML file is **C:\a\b\document.htm**, the string before the 3rd slash is replaced with http://x/y/ and the document is imported with the document reference **http://x/y/document.htm**.

ImportRenderedHTMLRefReplaceCSVs

Allows you to specify which parts of a rendered HTML file's reference (contained in the **ImportRenderedHTMLFieldName** field) you want to replace. Enter one or more strings that you want to be replaced with strings specified in **ImportRenderedHTMLRefReplaceWithCSVs**.

If you specify multiple strings, you must separate the individual strings with a comma (there must not be any space before or after the comma).

For example:

ImportRenderedHTMLRefReplaceCSVs=c:\inetpub\wwwroot

ImportRenderedHTMLRefReplaceWithCSVs=http:\\www.company.com

In this example, if a HTML file's reference is **c:\inetpub\wwwroot\docs\example.html**, the string **c:\\inetpub\wwwroot** is replaced with **http:\\www.company.com**, so that the document's reference is imported as **http:\\www.company.com\docs\example.html**.

ImportRenderedHTMLRefReplaceWithCSVs

Enter one or more strings with which you want to replace the strings that you have specified for **ImportRenderedHTMLRefReplaceCSVs**.

For example:

ImportRenderedHTMLRefReplaceCSVs=c:\inetpub\wwwroot

ImportRenderedHTMLRefReplaceWithCSVs=http:\\www.company.com

In this example, if a HTML file's reference is **c:\inetpub\wwwroot\docs\example.html**, the string **c:\\inetpub\wwwroot** is replaced with **http:\\www.company.com**, so that the document's reference is imported as **http:\\www.company.com\docs\example.html**.

Note:

You can use the following wildcards for **ImportRenderedHTMLRefReplaceCSVs** if you want to add the **ImportRenderedHTMLRefReplaceCSVs** strings to a rendered HTML file's reference instead of replacing it:

^ adds a string to the beginning of the HTML file's reference.

$ adds a string to the end of the HTML file's reference.

For example:

ImportRenderedHTMLRefReplaceCSVs=^,$

ImportRenderedHTMLRefReplaceWithCSVs=http://www.,.com

In this example the string **http://www.** is added to the beginning of the HTML file's reference, and the string **.com** is added to the end of the HTML file's reference. (That is, if the file's reference is **autonomy**, it is imported as **http://www.autnomy.com**).

Importing PDF files

The following settings allow you to determine if PDF files are eligible for importing, and how they should be opened.

ImportPDFUsePDFOpen

Note: if you are using Adobe Acrobat 4.0, you don't have to specify **ImportPDFUsePDFOpen**, as you can use a reference of the form **<PDFURL>#Page=<PAGENUMBER>**.

Enter **true** if you want to use a **pdfopen.asp** or **pdfopen.jsp** script to open a PDF document directly at a page that contains relevant information when you query IDOL server. Use **ImportPDFOpenPath** to specify the location where you have stored the script. (The scripts are available on request.)

If you enter **false** the PDF document will be opened at the first page. This is the default.

ImportPDFOpenPath

Note: if you are using Adobe Acrobat 4.0, you don't have to specify **ImportPDFUsePDFOpen**, as you can use a reference of the form **<PDFURL>#Page=<PAGENUMBER>**.

Enter the location of the **pdfopen.asp** or **pdfopen.jsp** script that is used when **ImportPDFUsePDFOpen** is set to **true**. (The scripts are available on request.) The scripts allow you to open a PDF document directly at a page that contains relevant information when you query IDOL server.

ImportPDFCheck

Enter **true** if you want to check if PDF files conform to the following settings (this is the default):

ImportPDFCheckMaxAverageWordLength

ImportPDFMaxCapitalRatio

ImportPDFMaxAverageASCII

ImportPDFMinAverageASCII

If you set **ImportPDFCheck** to **true**, you can use **ImportPDFCheckPDFDelete** to decide whether PDF files that do not conform to the specified settings should be imported or not.

Enter **false** if you do not want to check if PDF files conform to the specified settings.

ImportPDFCheckPDFDelete

If you have set **ImportPDFCheck** to **true**, **ImportPDFCheckPDFDelete** allows you to specify if PDF that do not conform to the following settings should be imported or not:

ImportPDFCheckMaxAverageWordLength

ImportPDFMaxCapitalRatio

ImportPDFMaxAverageASCII

ImportPDFMinAverageASCII

Enter **true** if you want to delete invalid PDF documents from the import folder (this is the default).

Enter **false** if you want to import all PDF documents, irrespective of whether they conform to the specified settings or not. If a document does not conform to the settings and you have set **ImportPDFCheckPDFDelete** to **false**, a log entry that states that this file is invalid is made when the file is imported.

ImportPDFCheckMaxAverageWordLength

If you have set **ImportPDFCheck** to **true**, you can use **ImportPDFCheckMaxAverageWordLength** to specify the maximum average length that words in a PDF file can have for the file to be accepted as valid. Enter the maximum number of characters that the average word can comprise. By default this is **8**.

For example:

ImportPDFCheck=true

ImportPDFCheckMaxAverageWordLength=8

In this example, if a PDF file contains words that on average longer comprise more than **8** letters, the document is regarded as invalid, and will not be imported if **ImportPDFCheckPDFDelete** is set to **true**.

ImportPDFMaxCapitalRatio

If you have set **ImportPDFCheck** to **true**, you can use **ImportPDFMaxCapitalRatio** to specify the percentage of capital letters that PDF documents can contain to be accepted as valid. By default this is **30**.

For example:

ImportPDFCheck=true

ImportPDFMaxCapitalRatio=30

In this example, if more than **30** % of the letters in a PDF file are capital, the document is regarded as invalid, and will not be imported if **ImportPDFCheckPDFDelete** is set to **true**.

ImportPDFMaxAverageASCII

If you have set **ImportPDFCheck** to **true**, you can use **ImportPDFMaxAverageASCII** to specify the average maximum ASCII value that the characters in a PDF document can have for the file to be accepted as valid. By default this is **120**.

For example:

ImportPDFCheck=true

ImportPDFMaxAverageASCII=120

In this example, if the characters in a PDF file on average have a higher ASCII value than **120**, the document is regarded as invalid, and will not be imported if **ImportPDFCheckPDFDelete** is set to **true**.

ImportPDFMinAverageASCII

If you have set **ImportPDFCheck** to **true**, you can use **ImportPDFMinAverageASCII** to specify the average minimum ASCII value that the characters in a PDF document can have for the file to be accepted as valid. By default this is **40**.

For example:

ImportPDFCheck=true

ImportPDFMinAverageASCII=40

In this example, if the characters in a PDF file on average have a lower ASCII value than **40**, the document is regarded as invalid, and will not be imported if **ImportPDFCheckPDFDelete** is set to **true**.

Importing BIF files

The following settings allow you to specify how BIF files are imported to IDOL server IDOL server.

ImportBIFReferenceField

Specify the name of the field in the BIF file that you want to import as the IDOL server reference field.

For example:

ImportBIFReferenceField=agentid

In this example, the value of the **agentid** field from a BIF file is imported as the value of a document's IDOL server reference field.

ImportBIFIncludeQuotes

ImportBIFIncludeQuotes allows you to specify whether double quote marks (**"**) from BIF file content are included in the data that the import module imports to IDOL server IDOL server.

Enter **true** to include double quote marks from the BIF file content.

Enter **false** to remove double quote marks.

Importing binary content

The following settings allow you to specify if binary content is imported into IDOL server and to define which type of binary content is eligible for importing.

ImportFilterBinaryContent

Enter **false** if you want to import any binary data that documents may contain. Otherwise enter **true**.

The default setting is **false**.

ImportMinAllowedAsciiCode

If you have set **ImportFilterBinaryContent** to **true**, you can use **ImportMinAllowedAsciiCode** to specify the minimum ASCII value allowed in characters in the binary content. Anything lower is treated as binary and removed.

The default setting is **0**.

ImportMaxAllowedAsciiCode

If you have set **ImportFilterBinaryContent** to **true**, you can use **ImportMaxAllowedAsciiCode** to specify the maximum ASCII value allowed in characters in the binary content. Anything higher is treated as binary and removed.

The default setting is **175**.

ImportMaxAverageWordLength

If you have set **ImportFilterBinaryContent** to **true**, you can use **ImportMaxAverageWordLength** to specify the maximum number of characters that binary words may contain. Any binary word that contains more than the specified number of characters will not be imported.

The default setting is **25**.

ImportMaxBinaryCharsPerHundred

If you have set **ImportFilterBinaryContent** to **true**, you can use **ImportMaxBinaryCharsPerHundred** to specify the maximum number of binary characters that are permitted per 100 characters. Any binary characters that exceed this number will not be imported.

The default setting is **3**.

Importing delimited files

Delimited information contains data fields that are delimited by certain separators. For example:

> ISBN-0749401161 Winnie the Pooh, 5 years old, main character of The house at Pooh corner, Andrew A Milne.

This content could be imported to obtain fields (in order):

> DREREFERENCE, DRETITLE, AGEGROUP, DRECONTENT, AUTHOR.

You can have multiple entries in one delimited file. The Import module is also flexible enough to deduce that if the start delimiter is not specified, the next field must be taken from the last end delimiter. This saves typing in duplicate delimiters.

Note:

- In order for any content to be imported, one of the fields that you import delimited information into must be DREREFERENCE.
- If you are importing from files that contain binary characters (which are not permitted in ASCII text editors/readers and so on), you can use the following control characters to allow the Import module to import binary characters correctly:

Character	ASCII code	Character	ASCII code	Character	ASCII code
<NUL>	0	<VT>	11	<SYN>	22
<SOH>	1	<FF>	12	<ETB>	23
<STX>	2	<CR>	13	<CAN>	24
<ETX>	3	<SO>	14	<EM>	25
<EOT>	4	<SI>	15	<SIB>	26
<ENQ>	5	<SLE>	16	<ESC>	27
<ACK>	6	<CS1>	17	<FS>	28
<BEL>	7	<DC2>	18	<GS>	29
<BS>	8	<DC3>	19	<RS>	30
<HT>	9	<DC4>	20	<US>	31
<LF>	10	<NAK>	21	<SP>	32

You can use these control characters in the following parameters:

ImportDelimitedDocStart

ImportDelimitedDocEnd

ImportDelimitedStart*n*

ImportDelimitedEnd*n*

For example:

String↵
text text text text
String↵
text text text text

In this example the delimited document contains binary characters (carriage returns) that act as delimiters. In order to import the text content correctly, you need to set the **ImportDelimitedDocStart** parameter to **String<CR>**.

The following settings allow you to specify how the Import module deals with delimited files.

ImportDelimitedExtns

Allows you to specify one or more file extensions that should be treated as delimited files when they are imported.

Note: If you want to specify multiple extensions, you need to separate them with commas (there must be no space before or after a comma).

For example:

ImportDelimitedExtns=*.html,*.dat,*.nfo

In this example, any **html**, **dat** or **nfo** file is treated as an delimited file during the importing process.

ImportDelimitedDocStart

Enter a string that marks the start of each record that you want to import from a delimited file. Any records that the file contains between the specified string and the **ImportDelimitedDocEnd** string are imported into IDOL server.

For example:

DocStart
Content
DocEnd
DocStart
Content
DocEnd

If a delimited file contains the above and you have specified the following settings, the **Content** records between **DocStart** and **DocEnd** are imported.

ImportDelimitedDocStart=DocStart

ImportDelimitedDocEnd=DocEnd

ImportDelimitedDocEnd

Enter a string that marks the end of each record that you want to import from a delimited file. Any records that the file contains between the **ImportDelimitedDocStart** string and the specified string are imported into IDOL server.

For example:

DocStart
Content
DocEnd
DocStart
Content
DocEnd

If a delimited file contains the above and you have specified the following settings, the **Content** records between **DocStart** and **DocEnd** are imported.

ImportDelimitedDocStart=DocStart

ImportDelimitedDocEnd=DocEnd

ImportDelimitedStart*n*

Enter a string that marks the start of the value that you want to store in the **ImportDelimitedField*n*** field. The end of the value is marked by the **ImportDelimitedEnd*n*** string.

For example:

Name:John Smith

Job title:IT Manager

E-mail:jsmith@company.com

Name:Pierre Martin

E-mail:pierremartin@company.com

If a delimited file contains the above and you have specified the following settings, the **Name** value is stored in the **ImportDelimitedField*n*** field.

ImportDelimitedStart0=Name:

ImportDelimitedEnd0=Job title,E-mail

ImportDelimitedEnd*n*

Enter one or more strings that mark the end of the value that you want to store in the **ImportDelimitedField*n*** field. The start of the value is marked by the **ImportDelimitedStart*n*** string.

If you want to enter multiple strings you must separate them with commas (there must be no space before or after a comma).

For example:

Name:John Smith

Job title:IT Manager

E-mail:jsmith@company.com

Name:Pierre Martin

E-mail:pierremartin@company.com

If a delimited file contains the above and you have specified the following settings, the **Name** value is stored in the **ImportDelimitedField*n*** field.

ImportDelimitedStart0=Name:

ImportDelimitedEnd0=Job title,E-mail

ImportDelimitedField*n*

Allows you to specify the name of the IDX field in which you want to store the value marked by **ImportDelimitedStart*n*** and **ImportDelimitedEnd*n***. Note that one of the fields you set up here must be DREREFERENCE.

If you are using a DRE version 3, you need to set up corresponding DRE fields, otherwise the **ImportDelimitedField*n*** fields will not be indexed into the DRE.

For example:

ProductID:MyProductNumber1

Product:MyProductName1

Description: Description of MyProductName1

If a delimited file contains the above and you have specified the following settings, the **MyProductNumber1** value is stored in the **DREREFERENCE** field and **MyProductName1** value is stored in the **ProductName** field.

ImportDelimitedField0=DREREFERENCE

ImportDelimitedStart0=ProductID:

ImportDelimitedEnd0=Product

ImportDelimitedField1=ProductName

ImportDelimitedStart1=Product:

ImportDelimitedEnd1=Description:

ImportDelimitedSkipChars*n*

Allows you to specify the number of characters that you want to skip from the beginning of the corresponding delimited field (specified by **ImportDelimitedField*n)*** when importing it.

For example:

ImportDelimitedField0=Book

ImportDelimitedSkipChars=7

In this example, if the delimited **Book** field contains the text **Title: Analytical Biochemistry**, the first **7** characters are skipped and the field is imported with the value **Analytical Biochemistry**.

ImportDelimitedFillInRefFromHTML

If you are importing a delimited document and you are unable to find a reference in this document you can set **ImportDelimitedFillInRefFromHTML** to **true** in order to import the document as an HTML file. This means that if a reference cannot be found in the document its file name is used as a reference.

If you don't want to import delimited documents as HTML enter **false**. This is the default.

Importing XML files

The Import module automatically tries to determine whether documents that are waiting to be imported are XML files (by checking if the document contains an xml version tag, for example, <?xml version=1.0>), and, by default, imports all data from an XML file's contents.

You can use the following settings to specify how the Import module imports XML documents.

ImportXMLExtns

By default the Import module automatically tries to determine whether documents that are waiting to be imported are XML files (by checking if the document contains an xml version tag, for example, <?xml version=1.0>), and imports them accordingly. If you want the Import module only to import files that have a specific extension as XML, **ImportXMLExtns** allows you to specify which extensions documents must have in order to be treated as XML files when they are imported.

If you want to specify multiple extensions you must separate them with commas (there must be no space before or after a comma).

For example:

ImportXMLExtns=*.xml,*.xhtml,*.sgml

In this example, all file that have the extension **.xml**, **.xhtml** or **.sgml** will be imported as XML files.

ImportXMLAttribute*n*

Allows you to specify XML attributes that you want to be imported.

For example:

An XML file contains the following:

<PRODUCT ID="9999"> Text1 More Text </PRODUCT>

You have set:

ImportXMLField0=myfield

ImportXMLSearchCSVTags0=PRODUCT

ImportXMLAttribute0=ID

ImportXMLEntryOnly0=True

In this example, the ID attribute **9999** is imported.

ImportXMLSearchTagsByAttribute*n*

If a document contains multiple instances of a tag and each of these instance has a different attribute, **ImportXMLSearchTagsByAttribute*n*** allows you to specify the individual attributes, so that each tags value can be imported correctly.

For example:

An XML file contains the following:

```
<item value=name>
  John Smith
</item>
<item value=company>
  United Adobigs Ltd
</item>
```

In this example, you need to set **ImportXMLSearchTagsByAttribute*n*** as follows in order to import the values **John Smith** and **United Adobigs Ltd**.

ImportXMLSearchTagsByAttribute0=name

ImportXMLSearchTagsByAttribute1=company

ImportXMLSearchCSVTags*n*

By default the Import module imports all data from an XML file's contents. If you only want to import data from specific tags, **ImportXMLSearchCSVTags*n*** allows you to specify which XML tags mark content you want to import.

You can enter multiple tags to specify a hierarchical level that the Import module uses to navigate to the level from which you want to retrieve data. Once the Import module has navigated to the lowest level (the last tag that you have listed), it will import the data between the start and end tags.

For example:

An XML file contains the following:

```
<document>
  <description>
  Text1
    <document>
    Text2
    </document>
  </description>
</document>
```

You have set:

ImportXMLSearchCSVTags0=description,document

In this example, the Import module navigates to the **document** tag that is contained within the **description** tag, and imports the **Text2** content (into the corresponding **ImportXMLField*n*** field).

If you set **ImportXMLSearchCSVTags0** to **document**, the Import module navigates to the first **document** tag that it can find and imports the **Text1** content (into the corresponding **ImportXMLField*n*** field).

ImportXMLField*n*

Specify the field in the IDOL server in which you want the content of that is imported from XML files to be stored (this is all data content , unless you have used **ImportXMLSearchCSVTags*n*** to restrict from which tags data is imported).

For example:

ImportXMLSearchCSVTags0=name

ImportXMLField0=DREREFERENCE

In this example, if an XML contains the tag "**<title>**The future of genetic engineering**</title>**", the text "**The future of genetic engineering**" will be stored in the IDOL server field **DREREFERENCE**.

ImportXMLEntryOnly*n*

Enter **true** if you only want to import content that has the same hierarchical level as the last tag specified in **ImportXMLSearchCSVTags*n*** (rather than importing content that is stored in lower hierarchical levels within this tag as well).

Otherwise enter **false** (this is the default).

For example:

An XML file contains the following:

```
<document>
  <description>
  Text1
    <document>
    Text2
        <description2>
        more text
        </description2>
    </document>
  </description>
</document>
```

You have set:

ImportXMLSearchCSVTags0=description,document

ImportXMLEntryOnly0=true

In this example, the Import module navigates to the **document** tag that is contained within the **description** tag, and imports the **Text2** content (into the corresponding **ImportXMLField*n*** field).

If you set:

ImportXMLSearchCSVTags0=description,document

ImportXMLEntryOnly0=false

In this example, the Import module navigates to the **document** tag that is contained within the **description** tag, and imports the **Text2** and the **more text** content (into the corresponding **ImportXMLField*n*** field).

ImportXMLScanAllTags*n*

If an **ImportXMLSearchCSVTags*n*** tag is contained multiple times in an XML file and contains different values, you can set **ImportXMLScanAllTags*n*** to **true** in order to store all the values in the **ImportXMLField*n*** (as a comma-separated list of values).

Enter **false** if you only want to extract the first value of an XML tag into the **ImportXMLField*n***. This is the default.

For example:

An XML file contains the following tags:

<Property FormalName="Keyword" Value="VARIA" />

<Property FormalName="Keyword" Value="VERKEER" />

<Property FormalName="Keyword" Value="MEDIA" />

You have set:

ImportXMLField0=DREKEYWORD

ImportXMLSearchCSVTags0=Property

ImportXMLEntryOnly0=true

ImportXMLAttribute0=Value

ImportXMLScanAllTags0=true

In this example, the Import module navigates to the **Property** tag and imports the **Value** attribute as a comma-separated list of values into the **DREKEYWORD** field, so that this field is imported with the values **Varia**, **Verkeer** and **Gerecht**.

If an XML file contains the tags

<Property FormalName="ContentType" Value="Original" />

<Property FormalName="Language" Value="nl" />

<Property FormalName="Desk" Value="BIN" />

you can only import the data into two fields by setting:

ImportXMLField0=DREVALUE

ImportXMLSearchCSVTags0=Property

ImportXMLEntryOnly0=true

ImportXMLAttribute0=Value

ImportXMLScanAllTags0=true

ImportXMLField1=DREFormalName

ImportXMLSearchCSVTags1=Property

ImportXMLEntryOnly1=true

ImportXMLAttribute1=FormalName

ImportXMLScanAllTags1=true

In this example, the Import module navigates to the **Property** tag, imports the **Value** attribute as a comma-separated list of values into the **DREVALUE** field and the **FormalName** attribute into the **DREFormalName** field. The **DREVALUE** field is imported with the values **Original**, **nl** and **BIN**. The **DREFormalName** field is imported with the values **ContentType**, **Language** and **Desk**.

ImportXMLSingleItemEntryTag

If an XML document contains multiple records (for example a list of customer details), **ImportXMLSingleItemEntryTag** allows you to specify the start and the end of the individual record items, so that each item will be imported.

Enter a string to define the start and the end of the items that you want to import.

For example:

```
<CUSTOMER>
    <KEY>3</KEY>
    <SNAME>Smith</SNAME>
    <FNAME>Peter</FNAME>
    <EMAIL>PeterSmith@example.com</EMAIL>
</CUSTOMER>
<CUSTOMER>
    <KEY>3</KEY>
    <SNAME>Miller</SNAME>
    <FNAME>Susan</FNAME>
    <EMAIL>SusanMiller@example.com</EMAIL>
</CUSTOMER>
```

The XML document in this example lists customers. Specify the following to import the details of each customer:

```
ImportXMLSingleItemEntryTag=CUSTOMER
```

ImportXMLStripTags*n*

Enter **true** if you want to strip any remaining XML tags from the text between the **ImportXMLSearchCSVTags*n*** tags when a file is imported.

Otherwise enter **false** (this is the default).

For example:

An XML file contains the following:

```
<document>
  <description>
  Text1
    <document>
    Text2
      <description2>
        more text
      </description2>
    </document>
  </description>
</document>
```

You have set:

ImportXMLStripTags0=True

ImportXMLSearchCSVTags0=description,document

ImportXMLField0=DREREFERENCE

In this example, the Import module navigates to the first document tag that it can find in the description tag and strips the tags from the content that it finds in this field (including hierarchically lower levels). It then imports the content **Text2 more text** in to the IDOL server field **DREREFERENCE** (rather than importing **Text2 <description2> more text </description2>**).

ImportXMLStripTagsNTimes*n*

When you import an XML document that contains HTML, it is assumed that the HTML will be escaped, therefore the HTML parsing is run twice by default (to return the file to HTML format and then to remove the HTML tagging).

Within an XML environment, this means that sometimes the HTML parsing may strip too many tags off a given XML field. If this is the case you can use **ImportXMLStripTagsNTimes*n*** to specify how many times you want to strip an XML field of html tags when it is imported.

If you have imported an XML field and it contains incorrect tagging change the value that you have set for **ImportXMLStripTagsNTimes*n*** and re-import the document.

For example:

If a file contains the following string, stripping the HTML the default number of times (twice) would have the wrong effect:

XML string is <Keyword>&ch-comm-axx</Keyword>

The XML is stripped, so that after the first pass of the HTML parser the string is reduced to the following:

&ch-comm-axx

This is the correct value. However, after the second pass of the HTML parser the value changes to an incorrect value:

-comm-axx

Specifying **ImportXMLStripTagsNTimes0=1** and re-importing the file corrects the above mistake.

ImportXMLPassThru

When you have specified **ImportXMLExtns**, **ImportXMLPassThru** allows you to specify that XML is imported with no processing other than checking that the XML is valid.

If you set **ImportXMLPassThru** to **true**, the XML contained in the file is imported directly (to an IDX file) without applying any processing such as field operations.

If you set **ImportXMLPassThru** to **false**, processing that you have set up is applied to the file's contents.

Importing email and email attachments

The following settings allow you to specify if email attachments are imported and whether they are saved in a specified directory.

ImportAttachments

Enter **true** if you want to import any email that may be attached to documents that you are importing.

Enter **false** if you do not want to import email that is attached to documents. This is the default.

ImportSaveAttachments

Enter **true** if you want any emails that may be attached to documents that you are importing to be saved in the **ImportSaveAttachmentDir** directory.

Enter **false** if you do not want to save emails that are attached to documents (this is the default).

ImportSaveAttachmentDir

Enter the path to the directory in which you want to save any emails that may be attached to documents that you are importing. By default this is **.** .

ImportRefEmlRefIsFileName

Note: you should only use this setting if you are using File System Fetch on eml files in an Exchange 2000 IFS.

Enter **true** if you want the DREREFERENCE to be the file name rather than the email UID.

Enter **false** if you want the DREREFERENCE to be the email UID. This is the default.

Importing web pages

The following setting allows you to specify the security type of secure web pages that you want to import:

ImportURLSecurityType

If you use SSL security, **ImportURLSecurityType** allows you to enter the security type of the URLs that you specify for the **ImportURL** field operation. Possible types are **SSL_V23**, **SSL_V2**, **SSL_V3**, **TLS_V1** and **SSL_V23**.

By default, **ImportURLSecurityType** is not set.

If you use a proxy server to access the internet, you should specify the following settings in order to import the URLs that you specify for the **ImportURL** field operation:

ImportURLProxyHost

Enter the IP address (or name) of the machine on which the proxy server you use to access the internet is running.

ImportURLProxyPort

Enter the port number for communication with the proxy server you use to access the internet.

ImportURLProxyUsername

Enter the user name to use to log on to your proxy server you use to access the internet.

ImportURLProxyPassword

Enter the password to use to log on to your proxy server you use to access the internet.

Importing zip files

The Import module can import files that are contained in a zip file (without having to unzip the file first).

The Import module unzips the files to the directory you specify with **ImportTempDir**, and then imports the files into IDX format, so that they can be indexed into IDOL server. The details of this stage of the import process for zip files depend on how you set **ImportZipUseSingleDREDoc** .

ImportZipUseSingleDREDoc=true

All the uncompressed files in the zip file are imported as sections of a single IDOL server document, with the following field:

IDOL server reference field

The value in the field links to a single IDOL server document that includes all imported files.

Since there is just one set of metadata for all the files that the zip file contains, operations that rely on the IDOL server reference relate to the original zip file, rather than to the individual files within the zip. The following examples illustrate this effect:

Example: Linking

When an application uses the IDOL server reference to link back to an original document, the document that is displayed is the original zip file. That is, although context or concept summaries are generated for the individual file from the zip file, when a user clicks on a query result, the zip file is displayed, rather than the individual file that is relevant to the query - and there is no way of identifying the appropriate individual file.

Example: Deduplication

When the IDOL server reference field is used to ensure that duplicate files are not imported, it is the zip file that the deduplication applies to. Zip files that have similar IDOL server references would be recognized as duplicates, but if an identical document is stored in 2 different zip files, no duplication is identified.

Note: a limitation of setting **ImportZipUseSingleDREDoc** to **true** is that document summaries and IDOL server metadata sometimes appear inconsistent with each other.

ImportZipUseSingleDREDoc=false

Each uncompressed file in the zip file is imported as a separate IDOL server document, with the following fields:

IDOL server reference field

The value in the field is postfixed with the name of the individual file from the zip file.

ZIPFILENAME field

The path to the original zip file (if your connector is File System Fetch, this is the full path to the original file; if you are using another connector, this is the full path to the temporary location from which the file was imported).

UNCOMPRESSEDFILENAME field

The name of the uncompressed file contained within the zip file. When **ImportZipUseSingleDREDoc** is set to **false**, this is the name of the individual file that you are importing.

FULLUNCOMPRESSEDFILENAME field

The relative path to an individual zip file contained within the original zip file. For example if an uncompressed file is imported from a zip file called **MySecondZip** which was located in another zip file called **MyFirstZip**, the field value could be the following:

MyFiles\ZippedFiles\MyFirstZip.zip:MySecondZip.zip

Every file is imported with its own metadata, which means that it is possible to access the individual files from the zip file. However, there are restrictions on how operations that usually rely on the IDOL server reference field can be performed, and operations that usually rely on the IDOL server reference field need to be replaced with operations on custom fields in order to be able to access the appropriate files. The following examples illustrate this effect:

Example: Linking

The IDOL server reference cannot be used to link to either the original zip file or the appropriate individual file. Instead, you can enable linking to files by setting up a custom field that contains the document name.

Example: Deletion

For some connectors, you can set **DeleteByField** to **true** in the connector's configuration file, which enables you to specify a field to check in order to determine if a document should be deleted. It is not possible to delete the contents of an original zip file by using a file's IDOL server reference, but you can use the IDOL server reference to delete individual files from the IDOL server.

Example: Deduplication

Since each of the files from the zip file is imported as a separate IDOL server document, it is not possible to check for duplicates of an original zip file. You can enable deduplication by IDOL server reference of an individual document.

Similarly, if an individual file is removed from the zip file, it will not be removed from the IDOL server when the file is re-imported, since there is no way of comparing the entire original zip file with the new zip file.

The following import parameters allow you to import files that are contained in a zip file (without having to unzip the file first).

ImportZipExpandFiles

Enter **true** if you want to import the files (restricted by **ImportZipExtractMustHaveCSVS**) that are contained in a zip file to the **ImportTempDir** directory. The files are stored in this directory until they are imported. This is the default value for this setting.

Enter **false** if you do not want to import the files that are contained in a zip file.

ImportZipExtractMustHaveCSVs

Enter one or more strings that the name of a file within the zip file must contain in order to be extracted.

If you want to specify multiple strings you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings that you specify.

For example:

ImportZipExtractMustHaveCSVs=documents/internal/*.doc,*.xls

In this example any documents in the zip file whose name contains the string **documents/internal/*.doc** or ***.xls** are extracted.

ImportZipExtractCantHaveCSVs

Enter one or more strings that the name of a file within the zip file must not contain in order to be extracted.

If you want to specify multiple strings you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings that you specify.

For example:

ImportZipExtractCantHaveCSVs=documents/internal/*.doc,*.xls

In this example any documents in the zip file whose name contains the string **documents/internal/*.doc** or ***.xls** will not be extracted.

ImportZipExtractUnknown

Enter **true** if you want to import a zip file that contains one or more files which have no extensions. The files are extracted from the zip in order to determine if they can be imported.

Enter **false** if the files in the zip that you want to import all have extensions. This is the default.

ImportZipUseSingleDREDoc

Enter **true** to import all the files that are contained in a zip file as a single IDOL server document. All the files from the zip file have the same metadata, and IDOL server references link to the original zip file. This is the default value for this setting.

Enter **false** to import all the files from the zip file to separate IDOL server documents. Each of the files from the zip file has its own metadata, and IDOL server references contain the name of the individual file, rather than linking to the zip file.

Importing PST files

The Import module can import Outlook items (appointments, contacts, notes, tasks, messages and attachments) contained in PST files.

Note: in order to import PST files, you should ensure that the **pstconv.dll** file is located in the directory specified by **ImportDefaultSlaveDirectory**.

The Import module extracts the content of PST files to the directory you specify with **ImportTempDir**, and then imports the individual files into IDX format so that they can be indexed into IDOL server. Each item extracted from the PST file is imported as a separate IDOL server document, with its own metadata, including the following fields:

IDOL server reference field

If you have set **ImportPSTSaveOriginalDocs** and **ImportPSTSaveOriginalAttachments** to **true**, the unique value in the field links to the individual item in the directory in which the import module stores the files for original items and attachments (this is the directory specified by **ImportPSTDirectoryForOriginalDocs**).

If you have set **ImportPSTSaveOriginalDocs** and **ImportPSTSaveOriginalAttachments** to **false**, the unique value in the field links to the file for the individual item in the original PST file location.

Type field

The Outlook item's type. This can be one of the following:

"Appointment"	"Task"
"Contact"	"Message"
"Note"	"Attachment"

ParentEmail field

If the item extracted from the PST file is an attachment, the **ParentEmail** field specifies the email item that it is an attachment for in the original PST file.

EntryID field

The unique Outlook entry ID for the individual item.

PSTFile

The path to the original PST file from which the item was extracted.

FolderPath

The Outlook path to the folder that contains the individual item.

In addition, other fields are included with the IDOL server document, depending on the file type. (The configuration parameters listed below allow you to specify file types that you want to import.)

The following import parameters allow you to import the contents of a PST file:

ImportPSTSaveOriginalAttachments

Enter **true** to save attachments contained in a PST file to the directory you specify with **ImportPSTDirectoryForOriginalDocs**.

Enter **false** if you don't want to save the attachments contained in a PST file. This is the default.

ImportPSTSaveOriginalDocs

Enter **true** to save appointments, contacts, notes, tasks and messages contained in a PST file to the directory you specify with **ImportPSTDirectoryForOriginalDocs**.

Enter **false** if you don't want to save the appointment, contacts, notes, tasks and messages contained in a PST file. This is the default.

ImportPSTDirectoryForOriginalDocs

If you have set **ImportPSTSaveOriginalAttachments** or **ImportPSTSaveOriginalDocs** to **true**, you should use **ImportPSTDirectoryForOriginalDocs** to specify the path to the location in which you want to save the PST contents.

ImportPSTLogging

Enter **true** if you want warnings and errors generated when the Import module extracts the contents of PST files to be included in the import log.

Otherwise, enter **false**. This is the default value for this setting.

ImportPSTProcessAppointments

Enter **true** if you want to import appointments contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import appointments contained in a PST file.

ImportPSTProcessAttachments

Enter **true** if you want to import attachments contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import attachments contained in a PST file.

ImportPSTProcessContacts

Enter **true** if you want to import contacts contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import contacts contained in a PST file.

ImportPSTProcessEmails

Enter **true** if you want to import messages contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import messages contained in a PST file.

ImportPSTProcessNotes

Enter **true** if you want to import notes contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import notes contained in a PST file.

ImportPSTProcessTasks

Enter **true** if you want to import tasks contained in a PST file. This is the default value for this setting.

Enter **false** if you don't want to import tasks contained in a PST file.

Creating back up files

The following import parameters allow you to create back up copies of IDX files that are created by the importing process.

ImportCopyIDXTo

If you want to create back ups of the IDX files that the importing process creates, use **ImportCopyIDXTo** to enter the full path for the directory in which you want to store back up files.

If you don't set **ImportCopyIDXTo** then no back ups are created. This is the default.

Note: you should not store other files in this directory, since they may get deleted when the disk is cleaned (if they are older than the age specified with **ImportCopyIDXToAge**).

ImportCopyIDXToFlags

Enter a number that specifies how back up files should be stored. You can make up this number by adding up some of the following numbers, as appropriate:

0	files are stored with the same name as the original IDX file (this is the default)
1	files are stored with the date appended to the start of the file name (the date is given in the format YYYYMMDD)
2	files are stored in a directory structure according to the import date (YYYY/MM/DD)
4	files are stored with the creation time appended to the start of the file name (the time is given as a number of seconds since 1st January 1970)

For example:

ImportCopyIDXToFlags=6

In this example back up files are stored in a directory structure according to their date, and the creation time is appended to the start of each file name.

ImportCopyIDXToAge

Enter a number of seconds after which documents are deleted from the back up directory. The age of a document is calculated from the creation time of the file.

Set **ImportCopyIDXToAge** to **-1** if you don't want back up files to be deleted. This is the default.

Using slaves

Applications that use the Import module generally use the importslave.exe, which in turn calls the following slaves in order to convert non-HTML/text files into HTML format:

Pdfslave	for PDF files
Flashslave	for SWF files; note that Flashslave will not extract text which is part of a graphic (e.g. labels within an image)
Omnislave	for DOC, XLS, PPT, RTF files
Binslave	for any other binary format; Binslave attempts to extract textual information stored amongst binary data (this will not work for encrypted information)

New slaves can easily be written by third parties.

For details of how to configure the individual slaves, please refer to the **Pdfslave**, **Omnislave** and **Binslave** appendices.

Note that you do not need to configure Flashslave, but if you are using HTTPFetch to retrieve SWF files, you need to set **FollowFlash** to **true** in the HTTPFetch configuration file. Please refer to your HTTPFetch documentation for details.

The following settings allow you to allocate slaves to specific file types, and to set the maximum amount of time that the Import module should take to convert files.

ImportRegisterExtnCSVs*n*

Enter one or more file extensions to specify which file types the corresponding **ImportRegisterExtnCSVs*n*** slave should convert to HTML format.

For example,

ImportRegisterExtnCSVs0=PDF,DOC

ImportRegisterExecutable0=Testslave

In this example, the **Testslave** converts **PDF** and **DOC** files to HTML format.

ImportRegisterExecutable*n*

Specify the import slave that you want to use to convert the corresponding **ImportRegisterExtnCSVs*n*** file types to HTML format.

For example:

ImportRegisterExecutable0=Testslave

ImportRegisterExtnCSVs0=PDF,DOC

In this example, the **Testslave** converts **PDF** and **DOC** files to HTML format.

Note: make sure you do not add the .exe extension of the slave when setting **ImportRegisterExecutable*n***.

ImportSlaveTimeOut

Allows you to specify the number of seconds that the importing process can wait for each document to be imported before it times out.

ImportKeyviewSlaveDirectory

Note: this setting is for backwards compatibility only (Import module version 2.2.2 and earlier).

Enter the path to the directory in which the KeyView slave and its DLLs are stored.

ImportDefaultSlaveDirectory

Enter the path to the directory in which the slaves and their DLLs are stored.

ImportUseFailoverSlave

Enter **true** if you want the Import module to use Binslave if Omnislave fails, or if a file type is unknown. This is the default value for this setting.

Otherwise, enter **false**.

Appendix A: Deprecated settings

The following settings have been deprecated and are only supported for backwards compatibility.

Deprecated	Replaced by
ImportCheckPDF	ImportPDFCheck
ImportCheckPDFDelete	ImportPDFCheckPDFDelete
ImportExpandZipFiles	ImportZipExpandFiles
ImportKeyviewSlaveDirectory	ImportDefaultSlaveDirectory
ImportUnicodeAutoDetectLanguage ImportUnicodeExtns ImportUnicodeOriginalEncoding ImportXToSJISExtns	ImportCharsetConvExtns ImportCharsetConvEncodingFrom ImportCharsetConvEncodingTo
ImportUsePDFOpen	ImportPDFUsePDFOpen
MaxAverageASCII	ImportPDFMaxAverageASCII
MaxAverageWordLength	ImportPDFCheckMaxAverageWordLength
MaxCapitalRatio	ImportPDFMaxCapitalRatio
MinAverageASCII	ImportPDFMinAverageASCII

Appendix B: PDFslave

The Import module uses PDFslave to convert PDF files into HTML format.

Configuring PDFslave

The settings that determine how PDFslave operates are contained in the PDFslave configuration file, which is located in the installation directory of your connector. You can modify these settings in order to customize the PDFslave, for example, to specify which type of parsing you want to use for Adobe files (by setting the **Mode** parameter).

Note that you should generally attempt to use the Adobe parser for PDF slaves. However, if you are using a platform other than Windows or Solaris, or encounter any problems with the Adobe parser (for example if accented foreign characters are not extracted or spaces between words are removed) you should use the Autonomy parser instead.

Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

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

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

Entering string values

If the value that you want to enter for a parameter that requires a string contains quotation marks, you must put the value into quotation marks and escape each quotation mark that the string contains by putting a slash in front of it.

For example:

FIELDSTART0="<font face=\"arial\"size=\"+1\"><b>"

Here the beginning and end of the string is indicated by quotation marks while all quotation marks that are contained in the string are escaped.

If you want to enter a comma separated list of strings for a parameter, and one of the strings contains a comma, you must indicate the start and the end of this string with quotation marks.

For example:

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

If any string within a comma separated list contains quotation marks, you must put this string into quotation marks and escaped the quotation marks in the string by putting a slash in front of them.

For example:

ParameterName="<font face=\"arial\"size=\"+1\"><b>",dog,bird,"wing,beak",turtle

Applying modifications to PDFslave's operation

New configuration settings only take effect once the connector service is stopped and restarted.

Configuration file sections

The PDFSlave configuration file contains the following section:

[Configuration]

[Configuration] section

Mode

Enter **Adobe** if you want to use an Adobe parser for parsing PDF files into HTML format. This is the default if are working on Windows or Solaris.

Enter **Alternative1** if you want to use an Autonomy parser. This is the default if you are not working on Windows or Solaris. **Note:** you can change this setting only on Windows or Solaris.

ExtractUnicode

If **Mode** is set to **Adobe**, you can use **ExtractUnicode** to specify whether you want to use a Unicode word finder to extract Unicode words from PDF files and to convert them into HTML format.

Enter **true** if you want to use a Unicode word finder.

Enter **false** if you do not want to use a Unicode word finder.

ShowFonts

If **Mode** is set to **Adobe**, you can use **ShowFonts** to specify whether you want to include additional font information in the HTML file.

Enter **true** if you do want to include additional font information.

Enter **false** if you do not want to include additional font information.

FilterNonPrintingCharacters

If **Mode** is set to **Adobe**, you can use **FilterNonPrintingCharacters** to specify whether you want to include non-printing characters from the PDF files when they are converted to HTML.

Enter **true** if don’t want to include non-printing characters.

Enter **false** if you do want to include non-printing characters. This is the default.

Rotate

If a PDF file's page setup is not Portrait, Rotate allows you to rotate the file so that it is imported correctly. Enter a number to specify by how many degrees you want to rotate the file (by default this is **0**). The file is rotated clockwise.

OutputUTF8

Enter **true** if you want to use convert output text to UTF8. This is necessary if the output text is in a double byte language (for example, Japanese or Chinese). Otherwise enter **false**.

OutputMetaUTF8

Allows you to specify whether to convert the PDF document's metadata to UTF8. By default, **OutputMetaUTF8** has the same value as you set for **OutputUTF8**. You only need to set **OutputMetaUTF8** if you know that the metadata and content of your PDF documents have different encodings.

Enter **true** to convert output PDF metadata to UTF8. This is necessary if the output text is in a double byte language (for example, Japanese or Chinese). Otherwise, enter **false**.

PreserveStreamOrder

Enter **true** if you want PDFslave to output content in the order it is stored in the PDF (for example, in order to keep column content or blocks of text together).

Enter **false** if you want to preserve the layout of the PDF content (that is, the content is output line-by-line as it would appear on the page). This is the default.

PositionFilePath

In order to enable highlighting, specify the full path to the directory in which PDFSlave stores position files (position files contain information about the position of words in the PDF file, which allows words to be located for highlighting when the document is displayed).

Example configuration file

```
[Configuration]
ShowFonts=false
ExtractUnicode=false
Mode=Adobe
OutputUTF8=false
Rotate=0
PositionFilePath=C:\Autonomy\Import\PositionFiles
```

Appendix C: Binslave

The Import module uses Binslave to convert binary files (other than PDF, doc, xls, ppt or rtf) into HTML format. Binslave attempts to extracts textual information stored amongst binary data. Note that this will not work for encrypted information.

Configuring Binslave

The settings that determine how Binslave operates are contained in the Binslave configuration file, which is located in the installation directory of your connector. You can modify these settings in order to customize Binslave according to your requirements.

Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

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

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

Entering string values

If the value that you want to enter for a parameter that requires a string contains quotation marks, you must put the value into quotation marks and escape each quotation mark that the string contains by putting a slash in front of it.

For example:

FIELDSTART0="<font face=\"arial\"size=\"+1\"><b>"

Here the beginning and end of the string is indicated by quotation marks while all quotation marks that are contained in the string are escaped.

If you want to enter a comma separated list of strings for a parameter, and one of the strings contains a comma, you must indicate the start and the end of this string with quotation marks.

For example:

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

If any string within a comma separated list contains quotation marks, you must put this string into quotation marks and escaped the quotation marks in the string by putting a slash in front of them.

For example:

ParameterName="<font face=\"arial\"size=\"+1\"><b>",dog,bird,"wing,beak",turtle

Applying modifications to Binslave's operation

New configuration settings only take effect once the connector service is stopped and restarted.

Configuration file sections

The Binslave configuration file contains the following sections:

[Default]

[<Extension>]

Note: the number of sections depends on the number of **[<Extension>]** sections that you specify.

[Default] section

This section contains the settings that Binslave uses by default for any file types that it processes (unless an **[<Extension>]** section has been specified for them, in which case the settings in this section override the default settings for this file).

AllowLanguages

Specify the languages for which you want Binslave to find multibyte character text. You can enter one or more of the following values:

unicode_chinese

chinese

unicode_japanese

japanese

unicode_korean

korean

unicode_greek

greek

If you want to specify multiple language types, you must separate them with commas (with no space before or after a comma).

For example:

AllowLanguages=unicode_japanese,japanese

MinWordLengthCJK

Specify the minimum number of characters that a multibyte language word must have for Binslave to add it to the HTML file. If you set **MinWordLengthCJK** to **0**, there is no minimum word length (this is the default value).

MaxWordLengthCJK

Specify the maximum number of characters that a multibyte language can have for Binslave to add it to the HTML file. If you set **MaxWordLengthCJK** to **0**, there is no maximum word length (this is the default value).

MinPhraseLength

Specify the minimum number of words that a phrase must contain for Binslave to add it to the HTML file. By default this is **4**.

MinWordLengthInPhrase

Specify the minimum number of characters that words in a phrase must contain for Binslave to add the phrase to the HTML file. By default this is **4**.

LocalChars

Allows you to specify one or more non-English characters that you want Binslave to recognize. This prevents Binslave from discarding words that contain the specified characters.

For example:

LocalChars=ä,Ä,ö,Ö,ü,Ü

In this example, German Umlaut characters have been specified for LocalChars. This means that Binslave will recognize any words that contain these letters.

StripChars

Enter one or more characters that you want Binslave to strip from the text that it converts. Specify multiple characters as a list, with no separators.

For example:

StripChars=abc

In this example, Binslave removes all instances of **a**, **b** and **c** from the text that it converts.

StripNumbers

Enter **true** if you want Binslave to strip all numbers when creating the HTML file.

Enter **false** if you do not want Binslave to strip any numbers when creating the HTML file. This is the default.

MaxPunctuationRepeatLength

Specify the maximum number of times that a punctuation mark can occur in succession for Binslave to add all of the successive marks to the HTML file.

If a punctuation mark occurs more than the specified number of times in succession, Binslave strips it down to one occurrence. By default this is **3**.

For example:

MaxPunctuationRepeatLength=3

In this example, if a document contains 4 successive commas, Binslave adds only one comma to the HTML file.

RemoveRepeatedPunctuation

Enter **true** if you do not want Binslave to add successive occurrences of punctuation marks to the HTML file. Binslave will then strip successive marks down to one occurrence. This is the default.

Enter **false** if you want Binslave to add all successive occurrences of punctuation marks to the HTML file.

For example:

RemoveRepeatedPunctuation=true

In this example, if a document contains 2 successive commas, Binslave adds only one comma to the HTML file.

DatePostfixes

Specify which date postfixes you want Binslave to recognize. If you want to specify multiple postfixes, you must separate them with commas (there must be no space before or after a comma).

The default setting is **st,nd,rd,th**.

For example:

DatePostfixes=st,nd,rd,th

In this example, **DatePostfixes** is set to English date postfixes. This allows Binslave to recognize dates in the format **1st**, **2nd**, **3rd**, **4th** and so on.

[<Extension>] section

You can define an **[<Extension>]** section for any file type that Binslave processes. You must name the section after the file extension of the file type.

Any settings that you specify in the **[<Extension>]** section override the settings in the **[Default]** section, but only for this file type.

For example:

[txt]

<Settings_for_text_files>

[ppt]

<Settings for PowerPoint files>

Example configuration file

```
[Default]
MinWordLength=2
MaxWordLength=20
MinPhraseLength=4
MinWordLengthInPhrase=3
MaxPunctuationRepeatLength=3
RemoveRepeatedPunctuation=true

[ppt]
ExtractWideChars=on
MinPhraseLength=3
MaxPunctuationRepeatLength=2
```

Stop lists for Binslave

Note: you can configure Binslave to use different stop lists for different file types. Autonomy supplies the following stop lists:

phraselist_qxd.dat	Stop list for Quark Express documents.
phraselist_ppt.dat	Stop list for PowerPoint documents.
phraselist_doc.dat	Stop list for Word documents. This is used by default for any file types that do not have a specific stop list created for them.

You can create your own stop list for a file type that you want to import in the directory that contains the **Binslave.exe** file. You must name the stop list file as follows:

phraselist_<file_type_extension>.dat

For example:

phraselist_txt.dat

This stop list is used when Binslave processes files with the extension **txt**.

Appendix D: Omnislave

Autonomy Omnislave is a plug-in module that allows you to extract data from various binary file formats, so it can be indexed into an Autonomy IDOL server This includes any textual data that is stored within the binary files.

Using IDOL server you can then retrieve the information that you are looking for automatically.

Omnislave is a replacement for the KeyView filters previously used by Import module versions 2.2.2 and earlier.

Configuring Omnislave

The settings that determine how Omnislave operates are contained in the Omnislave configuration file, which is located in the installation directory of your connector. You can modify these settings in order to customize Omnislave according to your requirements.

Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

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

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

Entering string values

If the value that you want to enter for a parameter that requires a string contains quotation marks, you must put the value into quotation marks and escape each quotation mark that the string contains by putting a slash in front of it.

For example:

FIELDSTART0="<font face=\"arial\"size=\"+1\"><b>"

Here the beginning and end of the string is indicated by quotation marks while all quotation marks that are contained in the string are escaped.

If you want to enter a comma separated list of strings for a parameter, and one of the strings contains a comma, you must indicate the start and the end of this string with quotation marks.

For example:

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

If any string within a comma separated list contains quotation marks, you must put this string into quotation marks and escaped the quotation marks in the string by putting a slash in front of them.

For example:

ParameterName="<font face=\"arial\"size=\"+1\"><b>",dog,bird,"wing,beak",turtle

Applying modifications to Omnislave's operation

New configuration settings only take effect once the connector service is stopped and restarted.

Configuration file sections

The Omnislave configuration file contains the following sections:

[Configuration]

[<file_format>]

[Configuration] section

This section contains general configuration settings, which apply to all **[<file_format>]** sections that you have defined underneath the **[Configuration]** sections.

MagicEnable

Enter **true** if you want Omnislave to work out the type of files that it imports automatically by inspecting the file's binary data (this is the default). This ensures that files which have an incorrect or no extension are imported correctly.

If you enter **false**, Omnislave uses the **OmniConvertExtns*n*** and **OmniConvertLibraryCsvs*n*** settings to determine how it should import each file. If a file has no extension Omnislave will not import it.

OmniConvertExtns*n*

Enter the extension of the ***n*** file type that you want Omnislave to convert. You need to define a **[<file_format>]** section for the formats, that you enter for **OmniConvertExtns*n***.

Note: you do not need to set **OmniConvertExtns*n*** if you have set **MagicEnable** to **true**.

OmniConvertLibraryCsvs*n*

Enter one or more dynamic link libraries that you want Omnislave to use to convert the file type that you have specified for **OmniConvertExtns*n***.

If you want to enter multiple dynamic link libraries you must separate them with commas (there must be no space before or after a comma).

Omnislave will attempt to convert the specified file type using the first dynamic link library listed. If this dynamic link library fails, it attempts to convert the file using the second dynamic link library listed and so on.

Note: you do not need to set **OmniConvertLibraryCsvs*n*** if you have set **MagicEnable** to **true**.

OmniConvertConfigSectionCsvs*n*

Enter the file formats for which you have defined **[<file_format>]** sections. If you have defined multiple sections, you must separate the section names with commas (there must be no space before or after a comma).

Logging

Enter **1** to enable logging or **0** to disable logging. By default this is **0**.

LogAppend

Enter **true** if you want to append to the log file every time the connector runs.

Enter **false** if you want to create a new log file every time the connector runs (this is the default).

LogMaxKBytes

The maximum size that the log file can reach (in kilobytes) before it is deleted or archived and a new log file is created. By default this is **100** kb.

[<file_format>] section

This section contains configuration settings that apply only to this **<file_format>**. If you specify a setting for this **<file_format>** that is also specified in the **[Configuration]** section, the setting in this section overrides it for this **<file_format>**.

You should define a **[<file_format>]** section for each of the file formats that you want Omnislave to convert.

Note: If you are writing your own Omnislave library, you can set additional parameters in the **[<file_format>]** section that uses your slave, and have your code read these settings from the configuration file.

OutputCharSet

The character set to which you want to convert **<file_format>** files. Enter one of the following:

ASCII	CYRILLIC_ISO
UTF8	THAI
UCS2	EASTERNEUROPEAN
GREEK	EASTERNEUROPEAN_ISO
GREEK_ISO	TURKISH
HEBREW	CHINESESIMPLIFIED
HEBREW_ISO	CHINESETRADITIONAL
ARABIC	KOREAN
ARABIC ISO	SHIFTJIS
CYRILLIC	JIS
CYRILLIC_KOI8	EUC

StopList

If you are using Omnislave to convert PowerPoint documents, you can specify the following stop list, which allows Omnislave to strip non-conceptual information from PowerPoint documents:

pptconv.dat

IncludeHeaderAndFooter

Enter **true** if you are using Omnislave to convert Excel documents, and you want to include the documents' header and footer in the data that is converted. This is the default.

Enter **false** if you want the header and footer to be excluded from Excel documents that you convert with Omnislave.

Example configuration file

```
[Configuration]
OmniConvertExtns0=*.doc
OmniConvertLibraryCsvs0=wordconv.dll,rtfconv.dll
OmniConvertConfigSectionCsvs0=MSWord,Rtf
OmniConvertExtns1=*.rtf
OmniConvertLibraryCsvs1=rtfconv.dll
OmniConvertConfigSectionCsvs1=Rtf
OmniConvertExtns2=*.xls
OmniConvertLibraryCsvs2=excelconv.dll
OmniConvertConfigSectionCsvs2=Xls
OmniConvertExtns3=*.ppt
OmniConvertLibraryCsvs3=pptconv.dll
OmniConvertConfigSectionCsvs3=Ppt
Logging=0
LogAppend=true
LogMaxKBytes=500
MagicEnable=true

[MSWord]
OutputCharSet=ASCII

[Rtf]

[Xls]
OutputCharSet=ASCII
IncludeHeaderAndFooter=false

[Ppt]
OutputCharSet=ASCII
StopList=pptconv.dat
```

Using Omnislave to convert foreign language documents

If you want to use Omnislave to convert foreign language documents, you need to copy one or more of the following files into the installation directory of your connector:

TB_arabic_ucs2.txt	for Arabic
TB_ascii_ucs2.txt	for ASCII
TB_chsimplified_ucs2.txt	for simplified Chinese
TB_chtraditional_chsimplified.txt	for traditional Chinese indexed as Simplified Chinese
TB_chtraditional_ucs2.txt	for traditional Chinese
TB_cyrillic_ucs2.txt	for Cyrillic
TB_cyrillickoi8_ucs2.txt	for Cyrillic KOI8
TB_easterneuropean_ucs2.txt	for Eastern European
TB_greek_ucs2.txt	for Greek
TB_hebrew_ucs2.txt	for Hebrew
TB_iso8859.txt	for all ISO languages
TB_korean_ucs2.txt	for Korean
TB_shiftjis_ucs2.txt	for Japanese
TB_thai_ucs2.txt	for Thai
TB_turkish_ucs2.txt	for Turkish

Upgrading to Omnislave

The following connector versions install Omnislave automatically:

Documentum Fetch	version **2.2.5** or greater
Exchange Fetch	version **2.2.1** or greater
FileNet Fetch	version **2.2.1** or greater
FTPFetch	version **2.2.2** or greater
HTTPFetch	version **2.2.4** or greater
Moreover Fetch	version **2.2.2** or greater
NNTP Fetch	version **2.2.2** or greater
Notes Fetch	version **2.2.6** or greater
ODBC Fetch	version **2.2.5** or greater
OmniFetch	version **2.2.1** or greater
Oracle Fetch	version **2.2.5** or greater
Pop3Fetch	version **2.3.2** or greater

Note: any Autonomy connectors that are not listed above install Omnislave automatically.

Files needed to upgrade non-Omnislave connectors

To upgrade a non-Omnislave connector to use Omnislave you need the following files:

Windows NT or 2000	**UNIX**
importslave.exe	importslave.exe
omnislave.exe	omnislave.exe
omnislave.cfg	omnislave.cfg
excelconv.dll	excelconv.so
pptconv.dll	pptconv.so
pptconv.dat	pptconv.dat
rtfconv.dll	rtfconv.so
wordconv.dll	wordconv.so

Upgrading non-Omnislave connector versions

To upgrade a connector to use Omnislave:

1. Stop your connector from running using the **Services** dialog (if you are using Windows NT or 2000) or a stop script (if you are using UNIX).
2. Navigate to the installation directory of your connector and replace the current **importslave.exe** file with the Omnislave **importslave.exe** file.
3. Copy the following files into your connector installation directory:

   omnislave.exe

   omnislave.cfg

   excelconv.dll or **excelconv.so**

   pptconv.dll or **pptconv.so**

   pptconv.dat

   rtfconv.dll or **rtfconv.so**

   wordconv.dll or **wordconv.so**
4. Restart your connector.

Writing your own Omnislave library

Your library must implement and export the functions **convInit**, **convShutDown**, **convConvert**. Save the DLL or SO file that you write in the same directory as the other Import module slave libraries.

convInit

```
int convInit(const char* szConfigFile, const char* szSection);
```

argument	description
szConfigFile	Relative path to the Omnislave configuration file.
szSection	The Omnislave configuration file section in which any extra settings that the library uses are made. **Note:** if you want to add any configurable options to your Omnislave library, you should add them to the **[<file_format>]** Omnislave configuration file section that contains settings for this conversion, and have your code read these settings from the configuration file. See **Configuring Omnislave** section for details.

convInit should contain the start-up code for the library. The function should return one of the values listed under **Return values**. convInit is called once, when the library is loaded.

convShutDown

```
int convShutDown(const char* szConfigFile, const char* szSection);
```

argument	description
szConfigFile	Relative path to the Omnislave configuration file.
szSection	The Omnislave configuration file section in which any extra settings that the library uses are made. **Note:** if you want to add any configurable options to your Omnislave library, you should add them to the **[<file_format>]** Omnislave configuration file section that contains settings for this conversion, and have your code read these settings from the configuration file. See **Configuring Omnislave** section for details.

convShutDown should contain the code that closes the library and frees any memory used. The function should return one of the values listed under **Return values**. convShutDown is called once, when the library is closed.

convConvert

```
int convConvert(const char* szInFile, const char* szOutFile, const char*
     szConfigFile, const char* szSection);
```

argument	description
szInFile	Full path to the file to read the data to be converted from.
szOutFile	Full path to the file to write converted data to.
szConfigFile	Relative path to the Omnislave configuration file.
szSection	Omnislave configuration file section in which any extra settings that the library uses are made. **Note:** if you want to add any configurable options to your Omnislave library, you should add them to the **[<file_format>]** Omnislave configuration file section that contains settings for this conversion, and have your code read these settings from the configuration file. See **Configuring Omnislave** section for details.

charConvert should contain the code that reads the content of **szInFile**, converts it to HTML, and writes the converted data to **szOutFile**. The output HTML can include custom fields that you want to index; please refer to the **Custom Omnislave library code example** for an example.

The function should return one of the values listed under **Return values**.

charConvert is called multiple times - once for each data file that the library converts to HTML.

Return values

The **convInit**, **convShutDown** and **convConvert** functions must return one of the following values:

0 CONV_SUCCESS

The function was successful.

-1 CONV_ERROR_FILE_NOT_FOUND

The **szInFile** was not found or could not be read.

-3 CONV_ERROR_OUTPUT_EXISTS

The **szOutFile** exists already, or could not be created for another reason.

-4 CONV_ERROR_INVALID_FORMAT

The data format could not be processed.

-5 CONV_ERROR

General error.

-6 CONV_ERROR_PASSWORD_PROTECTED

The **szInFile** is protected by a password.

Custom Omnislave library code example

This example shows a very simple custom Omnislave library. The conversion that charConvert performs just includes writing the following HTML to the output file:

<html><head><META NAME="Company" CONTENT="Autonomy"></head></html>

(Your conversion code should read the content of the **szInFile** file, convert it to HTML and write it to the **szOutFile** file.)

```
#define CONV_SUCCESS       0

int convConvert(const char *szInFile,
          const char *szOutFile,
          const char *szConfigFile,
          const char *szSection)
          {
             FILE *f = fopen(szOutFile, "w");
             char *szFixedFieldName = "Company";
             char *szFixedFieldValue = "Autonomy";
             fprint(f, "<html>");
             fprint(f, "<head>");
             fprint(f, "<META NAME=\"%s\" CONTENT=\"%s\",
             szFixedFieldName, szFixedFieldValue);
             fprint(f, "</head>");
             fprint(f, "</html>");
             fclose(f);
             return CONV_SUCCESS;
          }

int convInit(const char *szConfigFile,
          constchar *szSection)
          {
             //put start-up code here
             return CONV_SUCCESS;
          }

int convShutDown(const char *szConfigFile,
          constchar *szSection)
          {
             //put clean-up code here
             return CONV_SUCCESS;
          }
```

Glossary

Connector

A connector is an Autonomy fetching solution (for example HTTPFetch, Oracle Fetch and so on) that allows you to retrieve information from any type of local or remote repository (for example, a database or a web site). It imports the fetched documents into IDX or XML file format and indexes them into IDOL server from where you can retrieve them (for example by sending queries to the IDOL server).

Database

An Autonomy database is an IDOL server data pool that stores indexed information. The administrator can set up one or more databases, and specifies how data is fed to the databases.

IDOL server (Intelligent Data Operating Layer server)

Autonomy software infrastructure is based on IDOL server which enables for organizations to process digital content automatically and allow applications to communicate with each other. It consists of data operations that integrate information by understanding content, and is therefore data agnostic.

IDX

Apart from XML files only files that are in IDX format can be indexed into IDOL server. You can use a connector to import files into this format or manually create IDX files.

Importing

After a document has been downloaded from the location it is stored in, it is imported to an IDX file format. This process is called "importing".

Indexing

After documents have been imported to IDX file format, their content (or links to the original documents) is stored in IDOL server. This process is called "indexing".

Query

You can submit a natural language query to the IDOL server which analyzes the concept of the query and returns documents that are conceptually similar to the query. You can also submit Boolean, bracketed Boolean and keyword searches to the IDOL server.

Index

G

H

I

L

M

O

P

Q

R

S

T

U