

## CloudView CV23 Connectors

# Table of Contents

Default Connectors.....	6
What's New?.....	7
About Connectors.....	8
Introduction.....	8
The Push API.....	8
Default and Optional Connectors.....	9
Creating a Standard Connector.....	9
Deploying Standard Connectors.....	9
Deploy a Connector Server.....	10
Define the Target Push API Server.....	10
Configure HTTPS.....	10
Deploying a Custom Connector.....	11
Controlling Connectors.....	11
Control Document Scans.....	11
Schedule Update Frequency.....	12
Setting up the Document Type Pushed by a Connector.....	12
Storing Documents in a Specific Data Model Class.....	13
Using Document Cache.....	13
Understand Typical Use Cases.....	13
Enable Document Cache.....	13
Change the Location of the Document Cache on the File System.....	14
Repush from Document Cache.....	14
Clear the Document Cache Entirely.....	14
Clear the Document Cache for a Specific Connector Only.....	14
Using Push API Filters.....	15
Defining an External Push API Server.....	15
Create a Custom Push API Server.....	15
Create a Source Linked to the External Instance.....	16
Scan Your Source and Check the External Instance.....	16
Using the Interconnector Service.....	17
Configure the Interconnector Server.....	18
Configure the File System Connector.....	18
Configure the JDBC Connector.....	18
Add an Interconnector Aggregation Processor.....	18
Scan Connectors.....	19
CSV Connector.....	20
Introducing the CSV Connector.....	20
Aggregate Column Values.....	20
Filter Columns.....	20
Customize Column Names.....	20
Configuring the CSV Connector.....	21
Configure the Connector.....	21
Check the CSV Config.....	23
Crawler Connector.....	24
Introducing the Crawler Connector.....	24
About the Crawler.....	25
Crawler Connector Architecture.....	25
Crawl Rules.....	25
Refresh.....	26
About Site Collapsing.....	29
URL Processing.....	29
URL Handling of Crawled Sites.....	30
About the Fetcher.....	31

About the HTTP Fetcher.....	31
Fetcher Authentication Process.....	34
Debug Authentication Problems.....	36
Configuring the Crawler.....	36
Configuring the Crawler.....	37
Define the Groups of URLs to Crawl.....	39
Define Crawl Rules.....	40
Specify the File Name Extension and MIME Types to Crawl.....	42
Sample Configurations for Different Use Cases.....	42
Configuring the Fetcher.....	44
Configure the Fetcher.....	45
Configure the Fetcher Headers.....	45
Crawl Secured Sites.....	45
Deploying the Crawler Connector.....	50
Specify the Crawler Server.....	50
Specify the Push API Server.....	50
Managing the Crawler Connector.....	51
Troubleshooting the Crawler Connector.....	51
Use the Crawler http.log.....	51
Dump the Crawler Repository.....	53
Performance Monitoring.....	54
Hardware Sizing.....	55
Determine the Hardware Requirements.....	55
Determine the Crawl Speed.....	56
Advanced Configuration.....	56
Crawl Rules.....	56
Crawl Rule Actions.....	58
How Priorities Work.....	60
Error Handling.....	61
Custom Configuration.....	61
JDBC Database Connector.....	63
Introducing the JDBC Database Connector.....	63
Prerequisites.....	63
Workflow.....	63
Installing JDBC Drivers.....	64
Installing Custom Code.....	64
Configuring the JDBC Database Connector.....	65
Connect to the JDBC Database.....	65
Specify the Query Parameters.....	66
Define the Fields to Crawl.....	72
Examples of JDBC Database Connector Configurations.....	76
Feed Fetcher Connector.....	81
Introducing the Feed Fetcher Connector.....	81
Configure the Connector.....	81
Files Connector.....	83
File Server Access and Security.....	83
Behavior.....	83
Limitations.....	83
Maximum Path Length.....	84
Local File Server Access.....	84
Remote File Server Access.....	84
FTP Server Access.....	85
HDFS Server Access.....	86
Security.....	86
Mount Drive at Service Startup.....	86
About the Files Connector Configuration.....	87
Filesystem Paths.....	87
MS Windows Specificity for HDFS File System.....	87
Exclude/Include Rules.....	87
Allowed Extensions.....	88

Search Nonindexed File Names.....	88
Configure the Files Connector.....	88
Basic Configuration.....	89
Advanced Configuration - Crawl HDFS Example.....	89
Maximize Performances.....	90
Maximize Crawl Throughput.....	90
Add a Metadata Compaction Preprocessor.....	90
Extending the Files Connector Through Plugins.....	90
Advanced Configuration Parameters.....	91
<b>IMAP Connector.....</b>	<b>96</b>
About IMAP Configuration.....	96
Subject Filters.....	96
Indexing Email Threads.....	96
Gmail.....	97
Subject Normalization.....	97
Configure the IMAP Connector.....	97
Configure the IMAP Server Global Configuration.....	97
Configure a User Account.....	98
Configure Folders.....	99
Check Connectivity.....	99
Folder Configuration Examples.....	100
Troubleshooting.....	101
Out of Memory Errors.....	101
Mails Do Not Group by Thread Subjects.....	101
<b>LDAP Connector.....</b>	<b>102</b>
About LDAP Configuration.....	102
Before You Start.....	102
LDAP Classes and Attributes.....	102
Add LDAP References.....	103
Classes to Index.....	104
Configure the LDAP Connector.....	104
Configure the Connection to the LDAP Server.....	104
Specify the LDAP Classes and Attributes.....	105
Parameter Descriptions.....	106
Global Configuration Parameters.....	106
Class Config Parameters.....	107
Attribute Parameters.....	107
<b>Logs Connector.....</b>	<b>109</b>
Logging Framework.....	109
log4j.....	109
Apache.....	109
Auto.....	110
Custom.....	110
Configure the Logs Connector.....	110
Configure the Connector.....	110
Check the Logs Connector Config.....	112
Use Case.....	112
Step 1: Analyze Log Levels.....	112
Step 2: Group Stack Traces to Refine Analysis.....	113
<b>Managed Push API Connector.....</b>	<b>115</b>
Introducing the Managed Push API Connector.....	115
Configuring the Managed Push API Connector.....	115
<b>RScan Client Connector.....</b>	<b>117</b>
Introducing the RScan Client Connector.....	117
Before You Start.....	117
RScan Global Architecture.....	118

Configuring the RScan Client Connector.....	118
<b>Replay Connector.....</b>	<b>120</b>
Introducing the Replay Connector.....	120
Configuring the Replay Connector.....	120
Deploy the Replay Server Role.....	120
Capture a Data Flow.....	122
<b>XML Connectors.....</b>	<b>124</b>
Introducing the XML Connector.....	124
Configuring the XML Simple Connector.....	124
Extract by XPath Method.....	125
Extracting by XML Elements Method.....	126
Extracting Using XSLT Method.....	127
Using the Split XML Documents Method.....	128
Property Descriptions.....	129
Configuring the XML Advanced Connector.....	131
About the Processor Pipeline.....	132
Configuring the PAPI Document Processor.....	132
Configuring the XML Element Processor.....	133
Configuring the XPath Processor.....	134
Configuring the XSLT Processor.....	134
Configuring the Tee Processor.....	134
Configuring the XML Attach Processor.....	134
Configuring the Child Split Processor.....	135
Configuring the XPath Split Processor.....	135
Configuring the Custom Processor.....	136
Developing a Custom XML Processor.....	136
Develop Regular and Split Processors.....	136
Compile and Deploy a Custom Processor.....	138
Installing Custom Code.....	138
Requirements.....	139
Install Custom Code.....	139

# Default Connectors

This guide describes the functionality, configuration, and deployment of the Exalead CloudView standard connectors, delivered by default with the product.

**Note:**

- The Lotus Notes connector requires a specific license.
- All standard connectors are created the same way. See [Creating a Standard Connector](#)
- All connectors except the Crawler connector are deployed and controlled the same way. See [Deploying Standard Connectors](#) and [Controlling Connectors](#).

## Audience

This document explains how to configure and deploy Exalead CloudView connectors. It is assumed that the reader has working knowledge of:

- The operating system on which the Exalead CloudView server and connectors are installed.
- Regular expressions

This document assumes that the reader is familiar with the Exalead CloudView Administration Console.

## Further Reading

You might need to refer to the following guides:

Guide	for more details on
Connector Programmer	connector customization.
Consolidation	data consolidation.

# What's New?

There are no enhancements in this release.

# About Connectors

This section introduces Exalead CloudView connectors and describes common procedures.

[Introduction](#)

[Creating a Standard Connector](#)

[Deploying Standard Connectors](#)

[Deploying a Custom Connector](#)

[Controlling Connectors](#)

[Setting up the Document Type Pushed by a Connector](#)

[Storing Documents in a Specific Data Model Class](#)

[Using Document Cache](#)

[Using Push API Filters](#)

[Defining an External Push API Server](#)

[Using the Interconnector Service](#)

## Introduction

Connectors operate at the beginning of the Exalead CloudView indexing process. They send documents coming from different data sources to the Push API server (PAPI server), through the PAPI protocol.

This chapter gives you the basics about Exalead CloudView connectors:

### The Push API

The Push API is the public document API that allows Exalead CloudView to index data from any source.

It supports the basic operations required to develop new connectors, both managed and unmanaged.

- A managed connector is a piece of code running within Exalead CloudView. Package it as a Exalead CloudView Plugin to deploy and configure it in Exalead CloudView.

Develop it in Java, using the Connectors Framework API available in:

- `<INSTALLDIR>\sdk\java-customcode` for V6R2014 and higher versions.



- `<INSTALLDIR>\sdk\cloudview-sdk-java-connectors` in previous versions.
- An unmanaged connector is an external component that sends data to Exalead CloudView using the Push API. You can develop an unmanaged connector in any language, either by using Exalead CloudView Push API clients (available in Java, C# and PHP), or by directly targeting the HTTP API.

Manage and deploy unmanaged connectors yourself, as Exalead CloudView is not aware of these connectors.

**Note:** For more details on the Push API and connector customization, see the Exalead CloudView Connector Programmer's Guide

## Default and Optional Connectors

This guide covers the configuration of the connectors installed by default with Exalead CloudView. You can also install optional connectors for other data sources, for example, Microsoft SharePoint, ENOVIA ER, EMC Documentum, etc. For more information, contact your Exalead representative.

## Creating a Standard Connector

Add a new connector for each source that Exalead CloudView must crawl.

1. In the Exalead CloudView, go to **Connectors** and click **Add connector**.
2. Enter a descriptive name for the connector.

**Note:** The name you give the connector is used (if configured) as a category in the **Category refinements** panel.

3. From **Creation mode**, select either **new** or **copy** if you want to copy an existing connector.
4. From **Type**, select the connector type, for example, **Database (JDBC)**.
5. From **Push to PAPI server**, select the target Push API server.
6. Click **Accept**.

You can then configure the newly added connector. See the *Configuring the Connector* sections of the following chapters.

## Deploying Standard Connectors

When you create a new connector, you must specify which Connector Server hosts it. By default, the Java connector server already exists, and new connectors are automatically associated with it.

If for any reason whatsoever, the connector server role has been deleted, you are prompted to create a new one when adding the new connector.

Several connectors, however, require a separate connector server. For example:

- Some connectors must run as a 32-bit process, for example, old versions of the ENOVIA connector. For such cases, you need to deploy a new connector server.
- Some connectors must use C# connector servers, for example the MS SharePoint connector.

## Deploy a Connector Server

1. In the Administration Console, go to **Deployment > Roles** and click **Add Roles**.
2. In the list of roles, select the **Connector Server** and click **Accept**.
3. In the new **Connector server** box, complete the fields:
  - Select the **Type**.
  - For **Instance**, enter a unique name. For example, `java1`.
  - To run this connector server as a 32-bit process, select **32-bit**.
4. Click **Apply**.

## Define the Target Push API Server

Exalead CloudView can use several Push API servers to index data. When adding a connector, you must select the target Push API server that will store its data. Default is **Build group bg0**.

1. On the Administration Console **Home** page, under **Connectors**, click the connector name.
2. Go to the **Deployment** tab.
3. For **Push to PAPI server**, select the appropriate Push API server from the list.
4. Click **Apply**.

## Configure HTTPS

You may need to use HTTPS to secure your connector server.

1. Edit `Deployment.xml` in `<DATADIR>/config`.
2. Add the following in the `<Role name="ConnectorServer">` section and replace `HTTPS_PORT` with your HTTPS port:
 

```
<RoleAttribute name="httpsPort" value="HTTPS_PORT"/>
```
3. Save the file.
4. Restart Exalead CloudView.

## Deploying a Custom Connector

You can create and deploy custom connectors for your data sources.

To load your custom code in Exalead CloudView, you must package it as a CVPlugin. For information, see "Packaging Custom Components as Plugins" in the Exalead CloudView Programmer's Guide.

## Controlling Connectors

### Control Document Scans

In the Administration Console, the **Home > Connectors** section provides status information and the following actions for the managed connectors.

Action	Description
<b>Scan</b>	<p>Performs the scan for this connector, full or incremental, depending on connector configuration and previous scan operations.</p> <p>Scan requests are processed one after the other. When a scan request is launched while another scan is already being processed, it is added to a queue persisted to disk. It will start once the current scan is finished.</p>
<b>Abort scan</b>	<p>Stops the scan for a specific connector.</p> <p><b>Note:</b> It does not delete the documents already processed.</p>
<b>Clear documents</b>	<p>Deletes all connector documents from the system and resets the connector state.</p>
<b>More actions</b>	<p>Most connectors include more actions than <b>Scan</b>, <b>Abort scan</b> and <b>Clear documents</b>. For example, you can have actions such as <b>Force rescan</b>.</p>

**Note:** These actions can also be found under **Index > Connectors > YOUR CONNECTOR > Operation**.

## Schedule Update Frequency

As you do not want to start scans manually in production mode, it is important to schedule the scan frequency of your connector.

You can choose to add multiple schedules when your connector supports different scan modes. For example, a schedule to execute a full heavy scan each night at 00:00 when employees are not at the office, and another schedule to execute incremental scans every hour during working hours. See your connector documentation to know which scan modes are supported.

1. In Administration Console **Home** page, click the connector name.
2. Go to the **Operation** tab of the connector.
3. Click **Add schedule**, and define:

Action	to specify...
<b>Scan mode</b>	The scan mode to schedule, for example <b>Full scan</b> .
<b>Schedule type</b>	<p>The schedule frequency. It can be <b>Once</b>, <b>Monthly</b>, <b>Weekly</b>, <b>Periodic</b> or <b>Cron</b>.</p> <p>For <b>Cron</b>, you must use a Quartz cron expression (not a Linux cron expression). These expressions are supported by both Windows and Linux OS. It must include 6 required fields for: seconds, minutes, hours, day of month, month, day of week; and optionally, a year field separated by a white space. For more information, see .</p>
<b>Starts and Select execution time</b>	The start date and time.

4. Click **Apply**.

## Setting up the Document Type Pushed by a Connector

Exalead CloudView licensing relies on a token count. A token allows you to index a certain quantity of documents. This quantity depends on document types that are specified for each connector.

1. In the Administration Console, go to **Index > Connectors**, and select a connector.
2. In the **Configuration** tab, for **License document type**, select the document type that the connector will push to the Exalead CloudView index.

## Storing Documents in a Specific Data Model Class

By default, all documents are sent to the default data model Document class.

You can however specify another data model class for your connector documents by specifying a class name in the **CONNECTOR NAME > Configuration > Store in data model class** parameter.

## Using Document Cache

When you have a slow connector, or want to accelerate indexing throughput, use a document cache. The document cache stores documents pushed by a connector, before any processing was performed on them.

The document cache lifecycle is the same as that of the index: when a commit is made to the index, everything in the document cache (as well as everything being processed by the index server or that has gone through the PAPI server) is saved to disk.

## Understand Typical Use Cases

Specifically, the typical use cases are:

- During development, source throughput is too low.
- In production, because the fetch (required for document fetch, thumbnail, and preview) latency is too high.
- To ensure incremental updates for certain features, that is, updating a document without repushing it entirely. For example, Mashup Builder's social features such as tags require the document cache. When a tag is added, it is stored in the Mashup storage and this triggers a repush from cache operation for the impacted document. This repush from cache allows a document processor to retrieve the tags that are used to enrich documents before indexing.

## Enable Document Cache

1. Enable document cache for the build group.
  - a. In the Administration Console, go to **Deployment > Push to PAPI server**.
  - b. Select a build group, for example `bg0`.
  - c. Select the **Document cache** option.

By default, the document cache is enabled on all connectors of the build group.

2. To control the caching per connector:
  - Go to the **Connectors > CONNECTOR NAME > Deployment** tab and disable/enable the **Store in document cache** property.
  - You can also open the `<DATADIR>\config\Connectors.xml` file and edit the `SourceCachingConfig` parameters of source connectors to specify whether to enable the cache, and the maximum and minimum cache size.
3. Apply your changes.

## Change the Location of the Document Cache on the File System

By default the document cache is stored in the `cache` subdirectory of the build group. Yet, if the document cache grows too big for the build group's file system (for example, when the build group is on an SSD), you can specify another storage location.

1. Stop Exalead CloudView.
2. Open the `<DATADIR>\config\BuildGroups.xml` file.
3. Edit the `DocumentCacheConfig` node to add the `path` attribute: `path="path/to/new/Document/cache/location"`.
4. To generate the configuration, run `<DATADIR>/bin/buildgct`.
5. To keep the default document cache storage state, move the original document cache directory to the new location specified in step 3.
6. Restart Exalead CloudView.

## Repush from Document Cache

1. Go to the **Home** page.
2. Under **Indexing**, click **More actions**.
3. Click **Repush**.

## Clear the Document Cache Entirely

1. Go to the **Home** page.
2. Under **Indexing**, click **Clear**.
3. Select the **Document cache for bg0** check box.
4. Click **Clear**.

## Clear the Document Cache for a Specific Connector Only

1. Go to the **Home** page, or select **Connectors > name > Operation**.

2. Click **Clear documents**.
3. Select the **Clear cache entries for this connector** check box.
4. Click **Accept**.

This clears documents from both the index and the document cache.

## Using Push API Filters

You can encapsulate the `PushAPI` class using different "Push API filters" to enhance or modify its behavior. These filters allow you to include buffering, logging capabilities, debugging and other features.

For example, to trace PushAPI filter operations (for example, document size, CPU time, etc.), select the **Profile push operations** option in the **Deployment** tab. It adds logging details to the log file in `<DATADIR>/run/connectors-java0/`.

For more information, see "Push API filters" in the Exalead CloudView Connector Programmer's Guide.

## Defining an External Push API Server

For deployment reasons, you may want to push document sources (that is, one or more connectors, a Replay Server or a Consolidation Server) on an external Push API server located on another Exalead CloudView instance.

### Create a Custom Push API Server

1. In the Administration Console, go to **Deployment > Push API Servers**.
2. Click **Add Push API server**.
3. Configure the external Push API Server you want to target with the following parameters.

Parameter	Description
<b>Name</b>	Name of the custom Push API Server in your source instance. For example, <code>mypapi</code>
<b>Hostname</b>	Hostname of the external target instance.
<b>Port</b>	Push API Server port of the target instance, for example, <code>&lt;BASEPORT&gt;+2</code>
<b>Use HTTPS</b>	Enables HTTPS for the communication between the Push API servers of your source and target instances.

Parameter	Description
	For more information, see "Enable HTTPS for the Push API".
<b>Push API version</b>	As of now, only V4 is supported.
<b>Seen as ...</b>	Source name to display on the target instance. If not defined, the source connector (or Replay Server, or Consolidation Server) name is used by default.
<b>Login</b>	Login of the source that pushes documents to the target instance.
<b>Password</b>	Password of the source that pushes documents to the target instance.

4. Click **Apply**.

## Create a Source Linked to the External Instance

The second step consists in creating the source linked to the external Push API Server. that is to say a connector, a Replay Server or a Consolidation Server.

This procedure shows how to create a connector that pushes documents to the external Push API Server.

1. Go to **Index > Connectors**.
2. Click **Add connector**.
  - a. In **Name**, enter the connector name, for example `Filesystem1`
  - b. For **Type**, select a connector type, for example, `files`.
  - c. For **Push to PAPI server**, select your custom external PAPI server, for example `mypapisrv`.
  - d. Click **Accept**.
3. Configure your connector.
4. Click **Apply**.

## Scan Your Source and Check the External Instance

1. Go to the **Home** page.
2. Under **Connectors**, click **Scan** for the source connector that must push documents to the external PAPI Server.



Home					
Use this page to manage indexing and monitor running processes for a selected host.					
<b>Connectors</b> ⓘ					
Name	Type	Status	Tasks	Documents	
default	Unmanaged (Push API)	n/a	0	0	
Filesystem1 ⓘ	Files	idle	0 ~	0 ~	Scan

Neither the **Tasks** nor the **Documents** columns display counts for pushed documents (they stay with 0~), and the **Status** remains **idle**.

- Open the Administration Console of your external Exalead CloudView instance, that is, use the hostname defined previously as follows: <HOSTNAME> : <BASEPORT>+1 (do not use the port of the external Push API server) .

The source connector displays on the **Home** page with the name specified in the **Seen as...** Parameter. The **Documents** column shows the count of documents pushed by the source instance.

Home					
Use this page to manage indexing and monitor running processes for a selected host.					
<b>Connectors</b> ⓘ					
Name	Type	Status	Tasks	Documents	
consolidation-cs0	Unmanaged (Push API)	n/a	0	7	
default	Unmanaged (Push API)	n/a	0	0	
mypapisrv	Unknown / Removed	n/a	0	1,452	

## Using the Interconnector Service

The aim of the Interconnector Service is to use a Exalead CloudView role hosting an Apache ActiveMQ broker that acts as a message bus between a master and a slave connector.

This procedure shows you how to deploy an Interconnector Server with a JDBC connector (master) sending file system paths, indexed by a File System connector (slave). In this example, JDBC and File System documents are merged using a dedicated aggregation processor.

**Note:** You can also customize your connectors with a dedicated SDK to use the Interconnector Service. For more information, see "Customizing Connectors to use the Interconnector Service" in the Exalead CloudView Connector Programmer guide.

## Configure the Interconnector Server

First deploy your Interconnector Server.

1. In the Administration Console, go to **Deployment > Roles**.
2. In **Data Integration**, select **Interconnector Server**.
3. Enter an instance name and a port (61616 is the standard ActiveMQ port).
4. Click **Apply**.

## Configure the File System Connector

Now configure your File System connector (slave).

1. Open your File System connector.
2. In the **Advanced** tab, enter the name of your Interconnector Server in **Interconnector server instance name**.
3. In the **Configuration** tab, select a new and empty directory.
4. Click **Apply**.

## Configure the JDBC Connector

Now configure your JDBC connector (master).

1. Open your JDBC connector.
2. In the **Configuration** tab > **Fields selection**, unfold the column in which file paths are stored.
3. Click **Add column processor**.
4. Select **Interconnector Service** and click **Accept**.
5. In the **Advanced** tab:
  - a. Add the name of your Interconnector Server in **Interconnector server instance name**.
  - b. Add the name of your File System connector in **Slave connector**.
6. Click **Apply**.

## Add an Interconnector Aggregation Processor

Now that your connectors are properly configured, you can add an Interconnector aggregation processor.

1. Go to **Index > Consolidation > [your\_consolidation]**.
2. In **Aggregation processors**, click **Add processor**.
3. Enter a name and select **Interconnector Aggregator Processor**.

4. Click **Accept**.
5. Configure your aggregation processor as follows:
  - a. **Copy the parts**: select the check box to copy the parts from the child document.
  - b. **Child document type**: enter `child`.
  - c. **Relation document type**: enter `relation`.
  - d. **Delete child document**: select the check box to delete the child document containing parts and metas once its data has been processed.
  - e. **Copy the metas**: select the check box to copy the metas from the child document.
  - f. **Path from parent to child**: enter `parent.rel`.
  - g. Specify the metas of the child document that must be filtered in **Meta names filter**.
  - h. Specify the parts of the child document that must be filtered in **Part names filter**.
6. Click **Apply**.

## Scan Connectors

Now that your aggregation processor is configured, you can run a scan.

**Note:** The master connector must be scanned first.

1. Go to **Index > Connectors**.
2. Click **Scan** for your JDBC connector.
3. Click **Scan** for your File System connector.

# CSV Connector

This section describes the functionality and configuration of the CSV connector

[Introducing the CSV Connector](#)

[Configuring the CSV Connector](#)

## Introducing the CSV Connector

This chapter describes the supported features of the CSV Connector and how to configure it using Exalead CloudView Administration Console.

Using the CSV connector you can:

### Aggregate Column Values

Accumulation is the aggregation of column values on multiple rows that represent a single document. The CSV Connector can dynamically aggregate data contained in a CSV file before it is indexed based on the connector's settings.

By default, accumulation occurs on consecutive rows that have the same document URI. You can override the accumulation behavior by specifying columns to group by in **Group by**. This is similar to setting the primary keys for databases. If you specify column **A** as the Grouping column, then rows 2 and 3 merge into a single document with 2 values: **y** and **z**. See the **Group by** parameters in [Configure the Connector](#).

	A	B
1	1	x
2	2	y
3	2	z
4	3	z

### Filter Columns

By default, the CSV connector crawls every column of the CSV file. You can filter out columns when configuring the **Columns selection** parameters in [Configure the Connector](#).

### Customize Column Names

You can rename a column name to a meta name used in the Exalead CloudView index. This is particularly useful for CSV files that do not contain a header, which generate column names 0, 1, 2, etc. See **Custom column names** in [Configure the Connector](#).

## Configuring the CSV Connector

This section covers how to create and correctly configure your CSV connector in Exalead CloudView.

This section implies that the connector has already been added. See [Creating a Standard Connector](#).

### Configure the Connector

1. On the Administration Console home page , under **Connectors**, select your CSV connector.
2. Specify the filesystems paths to crawl
3. Configure the connector's main parameters as follows:

Parameter	Description
<b>Encoding</b>	Specifies the file encoding to use, by default, <b>UTF-8</b>
<b>Column delimiter</b>	Specifies the column delimiter to use. By default, commas ',' are used.
<b>Escaping character</b>	Specifies the character used to escape special characters. You can use it before the quoting character, the column delimiter, and the escaping character itself. Use \0 if you do not want to escape characters.
<b>Quoting character</b>	Specifies the characters used to begin and end a string. Use \0 if you do not want to index quoting characters.
<b>Treat first row as header</b>	Specifies if the columns of the first row of your CSV file are column headers. These column headers are used as meta names. If you do not have headers in your CSV file, clear this option. Column names are then imported as 0, 1, 2, 3. You can customize these names using the <b>Custom column names</b> option.
<b>File extensions</b>	Specifies the file extensions to process, by default, <b>csv</b> . Separate extensions with spaces.
<b>Filesystem paths</b>	Enter the filesystem path for the csv file or directory to crawl. You can prefix paths by: <ul style="list-style-type: none"> <li>• <code>data://</code> OR <code>/data</code> – relative to the <code>&lt;DATADIR&gt;</code></li> <li>• <code>resource://</code> – relative to the <code>NGRESOURCEPATH</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li>• <code>kit://</code> – relative to the <code>&lt;INSTALLDIR&gt;</code></li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li><code>file://</code> – absolute path, for example, <code>file://temp/csv-data/</code></li> <li><code>run://</code> – relative to the <code>NGRUNDIR</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li><code>config://</code> – relative to the <code>NGCONFIGDIR</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li>Click <b>Add item</b> for each path you want to crawl.</li> </ul>

#### 4. Configure the optional parameters:

Parameter	Description
<b>Max. row</b>	Specifies the maximum number of rows to crawl
<b>Prefix</b>	Specifies the prefix to add to each meta.
<b>Meta normalization</b>	Normalizes meta names by converting names to lowercase and spaces to underscores.
<b>Public security</b>	Pushes the <code>Everybody</code> security token with the document.
<b>Custom URI</b>	<p>Specifies the separator and the list of columns to concatenate and build the URI. Beware, in case of multivalued metas, only the first value is used.</p> <p>If empty, the URI is generated automatically as the file name plus the row number.</p>
<b>Columns selection</b>	<p>By default, the connector crawls all CSV file columns unless you specify a filter. You can specify a filter here to exclude or include columns, for example, column <b>D</b>.</p> <p><b>Note:</b> If the <b>Treat first row as header</b> option is selected, you can click in the <b>Columns</b> field to get column name suggestions.</p>
<b>Custom column names</b>	To rename the specified <b>Column name</b> to a <b>Meta name</b> that is used in the Exalead CloudView index.
<b>Group by</b>	<p>The set of columns included in the document URI determines how the rows are accumulated. For more information, see <a href="#">Aggregate Column Values</a>.</p> <p>The <b>Distinct</b> option ensures that if column values are identical, the value appears only once.</p>

Parameter	Description
<b>Push API filter</b>	Create an entry for each filter to apply. For more information, see "Push API filters" in the Exalead CloudView Connector Programmer's Guide

## Check the CSV Config

1. From the Exalead CloudView home page, click the connector name.
2. Click **Check config**.
3. Click **Apply** to save and apply the configuration changes.

You are now ready to scan and index your documents.

# Crawler Connector

This chapter describes how to configure the Crawler connector.

[Introducing the Crawler Connector](#)

[About the Crawler](#)

[About the Fetcher](#)

[Configuring the Crawler](#)

[Configuring the Fetcher](#)

[Deploying the Crawler Connector](#)

[Managing the Crawler Connector](#)

[Troubleshooting the Crawler Connector](#)

[Hardware Sizing](#)

[Advanced Configuration](#)

[Custom Configuration](#)

## Introducing the Crawler Connector

The Crawler connector can crawl any number of URLs if they are accessible from the server on which Exalead CloudView is installed.

This chapter describes how to configure the Crawler connector using Exalead CloudView Administration Console.

The crawler uses a **fetcher** to fetch URLs, submits them to the build chain, and extracts links. There is a default fetcher configuration used for crawling URLs. Independent fetcher parameters control the behavior of the fetcher. For more information, see [Configuring the Fetcher](#).

Before you configure your Crawler connector specific parameters:

- First add a connector as described in [Creating a Standard Connector](#).
- Define the behavior of the connector and ask yourself:
  - What do I want to crawl? Sites? Internet? Document types?
  - Do I need to filter URLs?
  - Do I need to define special behavior for certain URLs?



**Important:** Try to limit the number of crawler connector instances. A crawler can crawl many sites and you can index documents in different sources using crawl rules. Identify whether to use smart refresh or manual refresh (for a finer granularity).

## About the Crawler

### Crawler Connector Architecture

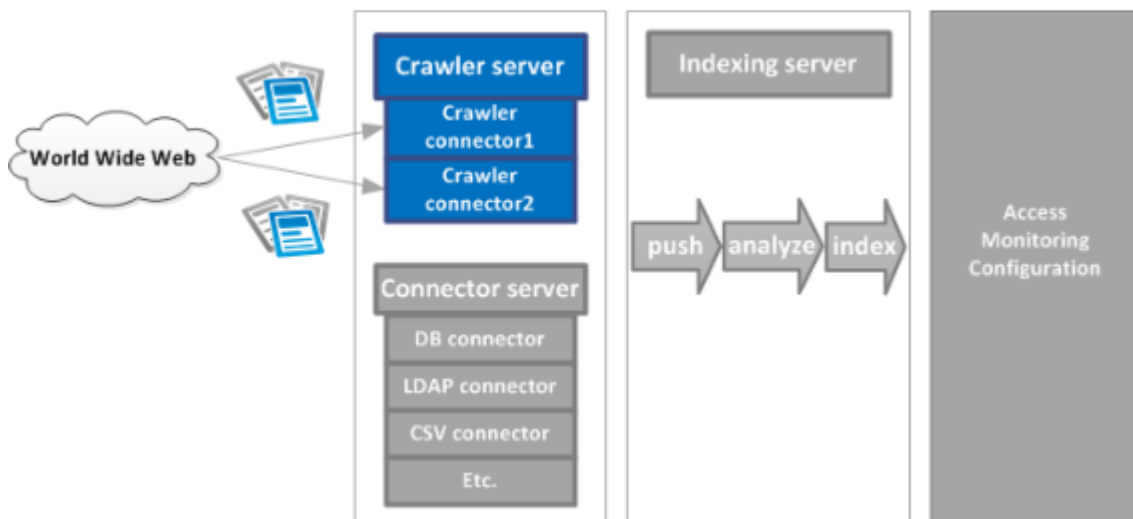
The global connector architecture has two different server processes for hosting the connectors:

- All connectors are hosted by the **Connector server**
- Except for Crawler connectors that are hosted by the **Crawler Server**.

The main components of the Crawler architecture are:

- Crawler Server
- Indexing server

#### *Global Crawler Architecture*



### Crawl Rules

Crawl rules are a set of patterns, matching URLs and assigning actions to them. The most important rules are those which define whether a URL must be crawled, indexed, and whether to follow the links found.

**Important:** Rule order is critical! All rules are always applied to each URL, starting with the first rule listed. The following rules can potentially override already-applied rules.

## Rule Types

When you specify a URL to crawl, you can define two main actions:

- The **index** rule allows you to index a URL,
- The **follow** rule allows you to follow its links.

The **index** and **follow** rules have their negative counterparts: **Don't index** and **Don't follow**.

An **ignore** rule can completely forbid a URL from being crawled.

**Note:** The URLs for which no crawl rules are specified, follow the default crawl rules defined in **General > Advanced options > Default behavior**.

## Rule Patterns

Matching can use a wide set of patterns. The most common pattern uses a regular expression on the whole URL string. Other pattern types include many shortcuts, matching only a component of the URL, or comparing a length.

Common patterns include:

- Whole URL prefix
- Host
- Domain
- File extension (suffix of path)
- Query parameter...

The patterns available in the Administration Console are prefixes on entire URLs, supporting regular expressions.

## Escaping

To avoid escaping special characters like `.` and reverse the escaping with `.`, use `\.`.

For example, to match all URLs on a site containing a `/path/` element, you can use the following pattern: `http://www.example.com/\.\/page/`.

## Refresh

By default, no refresh is enabled in a new crawler.

There are two modes to refresh crawled documents:

- Smart refresh

- Deterministic refresh

**Recommendation:** Use the Smart refresh mode. Deterministic refresh is generally only useful when crawling an intranet site with a controlled refresh rate, or when a specific latency is required.

## Smart Refresh

In **Smart refresh** mode, the crawler picks URLs from its repository depending on their age and update frequency. Fast-changing URLs are refreshed earlier than static ones.

There are two additional parameters: **Min** and **Max age**:

- **Min age** is the age below which a URL is never refreshed,
- and **Max age** is the age above which a URL is always refreshed.

For example, in the default configuration, a really fast-changing URL is never refreshed faster than every hour, and a completely static page is still refreshed every day.

These parameters assume that the crawler has the available computing and network resources to achieve this refresh, else it will do at best. The **Smart Refresh** mode is therefore a good choice when you do know the update frequency of documents.

## Deterministic Refresh

The **Deterministic refresh** mode is available through a MAMI command accessible by `cvcommand`.

**Note:** For more information about `cvcommand`, see "Get started with command-line interfaces" in the Exalead CloudView Administration Guide.

The command is called `RefreshDocuments` and can take two parameters:

- `maxAgeS`: to select only the URLs that have not been refreshed for a period of time (in seconds),
- `prefix`: to select only the URLs matching a prefix. This prefix is not a regular expression.

The command extracts all matching URLs from the crawler storage, and adds them in the fifos (the persistent structure containing the URLs to crawl). You can use the product scheduler to run this command periodically.

**Note:** This refresh can only be guaranteed if the crawler has enough computing and network resources to crawl all refreshed URLs before the next refresh. The number of URLs to refresh on a given host and the crawl delay (2.5 seconds by default) can also limit the refresh rate.

In the following example, the command refreshes all pages that are older than a day on `example.com`:

```
./bin/cvcommand :$MAMIPORT /mami/crawl refreshDocuments crawlerName=$CRAWLER
maxAgeS=86400 prefix=http://www.example.com/
```

Output in run/crawler-exa0/log.log:

```
[2019/07/11-15:52:34.164] [info] [exa.bee.BeeJob 1127] [crawler.feedfetcher]
Refreshing 8756 out of 10000 documents in prefix <http://www.example.com/>
[2019/07/11-15:52:38.781] [info] [exa.bee.BeeJob 1127] [crawler.feedfetcher]
Refreshing 17216 out of 20000 documents in prefix <http://www.example.com/>
[2019/07/11-15:52:49.790] [info] [exa.bee.BeeJob 1127] [crawler.feedfetcher]
Refreshing 25782 out of 30000 documents in prefix <http://www.example.com/>
...
[2019/07/11-15:54:35.684] [info] [exa.bee.BeeJob 1127] [crawler.feedfetcher]
Refreshed 145344 out of 172549 documents in prefix <http://www.example.com/>
```

**Note:** In this log, the last message means that the URLs have been scheduled for refresh but are not necessarily refreshed yet.

The URLs to refresh are added to a fifo, by default, it uses the highest priority. See [How Priorities Work](#).

You can check the refresh progress in:

- The Monitoring Console (in **<HOST> > Services > Exalead > Connectors > CRAWLER NAME**)
- The **Crawler connector > Operations & Inspection** tab, if you observe the `source.0.size` crawler probe values.

You can also run a deterministic refresh job periodically (at fixed times, dates or intervals), by editing the `<DATADIR>/config/Scheduling.xml` file, and specifying a Quartz cron expression. Exalead CloudView executes the MAMI commands defined in this xml file automatically.

In the following example, the `refresh_crawler` command uses a cron expression that runs the crawler refresh at "`0 0 0 * * ?`", meaning daily at midnight.

You can specify several schedules by adding as many `<TriggerConfigGroup>` as required.

```
<master:SchedulingConfig xmlns="exa:com.exalead.mercury.mami.master.v10">
<master:JobConfigGroup name="refresh_crawlers">
<master:DispatchJobConfig name="refresh_crawler">
<master:DispatchMessage xmlns="exa:exa.bee" messageName="refreshDocuments"
serviceName="/mami/crawl">
<master:messageContent>
<master:KeyValue key="crawlerName" value="CRAWLERNAME" />
<master:KeyValue key="maxAgeS" value="3600" /> <!-- optional, don't refresh
documents younger than 1 hour, else all documents regardless of age -->
<master:KeyValue key="prefix" value="URLPREFIX" /> <!-- optional, only
refresh documents with URL beginning with prefix, else all documents -->
```

```

</master:messageContent>
</master:DispatchMessage>
</master:DispatchJobConfig>
</master:JobConfigGroup>
<master:TriggerConfigGroup name="exa">
<master:CronTriggerConfig cronExpression="0 0 0 * * ?"
misfireInstruction="do_nothing"
jobName="refresh_crawler" jobGroupName="refresh_crawlers"
name="refresh_crawlers" calendarName="dummy_calendar" />
</master:TriggerConfigGroup>
</master:SchedulingConfig>

```

## About Site Collapsing

Often a site does not have a lot of rich content about a specific topic. If a Exalead CloudView user searches for that topic, the search results are overwhelmed with many pages from that site. The result is that other sites with important relevant information get crowded out.

The idea of site collapsing is to treat all the pages from a site as a single page. In this case, Exalead CloudView tries to select the most relevant page from the site, and only shows that one page in the search results. Other sites appear higher in the search results.

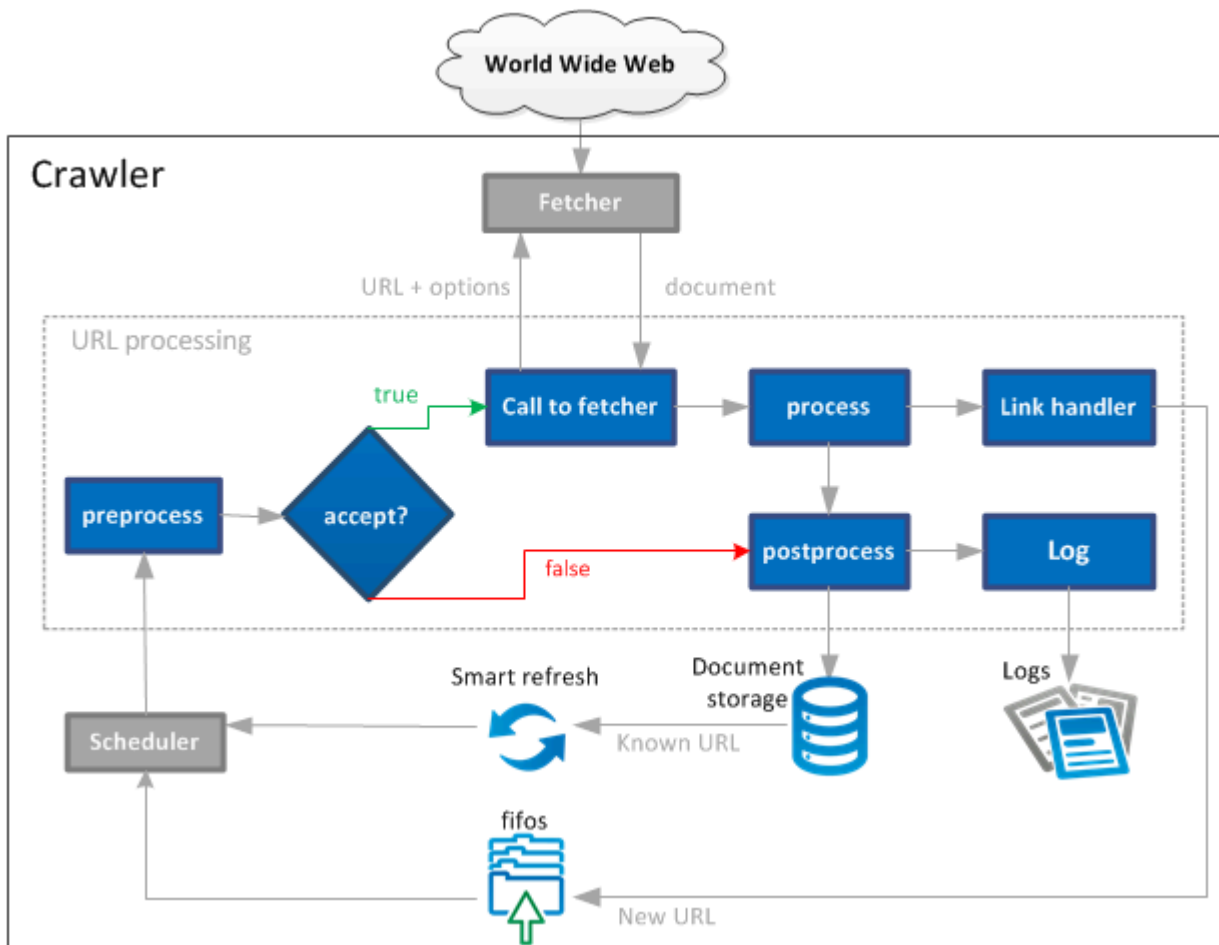
Site collapsing works on the assumption that if the page selected to represent the site does not contain the content a user is looking for, there must be enough good links (navigation menus) within the site to allow users to quickly find what they are looking for.

By default, the site collapsing ID is based on the host part of the URL. All URLs belonging to the same host share the same ID and are shown as one collapsed result. You can configure the site ID to include more segments of the URL's path to distinguish sites on the same host. You can do so by specifying the recursion with the **Depth** parameter. By default, the value is 0, which corresponds to the host part of the URL.

This option pushes a meta called `site_id` with the documents to index. This meta is based on the site extracted from a URL. To enable site collapsing, you also need to create a group in **Search Logics > Sort & Relevance** and in the **Expression** field, enter `site_id`.

## URL Processing

### *URL Processing in the Crawler*



## URL Handling of Crawled Sites

The Crawler connector has the following URL handling methods or constraints:

- Only HTTP and HTTPS protocols are accepted.
- Symbolic host names are syntactically checked and converted to lowercase; any trailing periods are removed.
- URL paths are checked for illegal characters (though "unwise" characters are accepted), unreserved characters are unescaped.
- When accepted, URL queries are checked for illegal characters (though "unwise" characters are accepted), unreserved characters are unescaped.
- Escaped characters are printed with lowercase hex digits.
- Fragment identifiers (#part) are checked for illegal characters, then removed.
- Literal IPV4 and IPV6 host names are supported, for example, `http://[2001:200:dff:fff1:216:3eff:feb1:44d7]/index.html`.

## About the Fetcher

This section describes how the HTTP fetcher works.

[About the HTTP Fetcher](#)

[Fetcher Authentication Process](#)

[Debug Authentication Problems](#)

### About the HTTP Fetcher

The HTTP Fetcher is the component allowing to download a document from its URL.

The crawler uses it, but also the product's generic fetcher, to generate previews, thumbnails, and direct downloads from the Mashup UI.

The fetcher configuration defined in the Administration Console > **CRAWLER connector** > **Fetcher** tab, is also used by the `fetchapi` remote API, called by the Mashup UI for downloads, thumbnails, and previews.

You can define different fetch configurations. By default, in the Administration Console, each new crawler is associated with a new fetcher configuration.

### HTTP Common Parameters

**Cookies** are always processed, but never stored by default. They are however automatically enabled, stored, and sent back on sites configured with an HTML form authentication. The global **Enable cookies for all sites** parameter allows you to store and send cookies for all sites. Only cookies following the standard privacy rules are stored, as web browsers do.

The **User agent** by default identifies itself as a Mozilla-compatible Exalead CloudView. When crawling the public internet, customize it with a URL pointing to a page describing the crawler and how to contact the administrator.

The **From** field is empty by default, and can contain a contact email address, which is useful when crawling the public internet.

You can add any HTTP **header** to the default request. The default configuration only contains `Accept: */*`, which is required by some sites.

### Conditional Fetch

The fetcher uses the `Last-Modified date` and `E-Tag` HTTP Headers to allow conditional fetch.

The crawler uses this condition when refreshing pages, to avoid fetching unchanged documents. In that case, the server only answers `304 Not Modified`.

## HTTP Authentication

The fetcher can access pages protected through different kind of authentication protocols.

- **Basic** only requires a username and a password. They are sent as plain text in the HTTP request.
- **Digest** requires a username, a password, and an optional realm. Unlike Basic, the Digest protocol avoids sending the password as plain text.
- **NTLM** requires a username, a password, and either a hostname or a domain.

Configure URL patterns to tell the fetcher which sites the login information applies to. They have the same syntax and meaning that the patterns used in the crawler configuration. They must match only the URLs where the login information is required.

**Important:** If you do not configure any URL pattern, the authentication applies to all URLs crawled. This can be dangerous, as the password might be sent to a third party.

## HTML Form Authentication

The fetcher also supports authentication using HTML forms. This applies to any website where the login procedure involves filling in an HTML form and submitting it through a `POST` request.

### What You Must Know

To properly configure the fetcher to authenticate using an HTML form, gather the following information:

- Find patterns to identify which URLs require authentication. Sometimes authentication is not required or not relevant to retrieve some parts of a website, for example static resources or images. URL patterns are the same than for standard HTTP authentication (see above), and the same recommendations apply. They work the same way as crawl rule patterns, and support the same syntax.
- Describe how to identify a successfully authenticated page from an unauthenticated one. This authentication test requires a **success/failure condition**, and is applied to all crawled URLs matching the above patterns. You can either choose a characteristic of the failed authentication page, to make a "failure condition", or a characteristic of a successfully authenticated page, to make a "success condition". See below for examples.
- Find the following information to authenticate:
  - First, find the URL containing the authentication form. This URL is called the **gateway URL**. It must be unique for the whole site. It is often the URL you are redirected to, when attempting to load a page without being authenticated.



- Then, find the authentication form in the HTML. Check, using "view source", that the form is statically embedded in the HTML and not dynamically generated using javascript. In that last case, the fetcher cannot find it. If it is in an iframe, use the iframe URL instead. If the form is not the first one in the page, tell the fetcher how to find the right one: you can use the element's name, id, or class as required. The first matching form is used.
- To specify the form input in the fetcher **Form fields**, you have to find the input name of each required parameter, usually a user name, and a password. In the following input form example, the `user` field contains the user name, and the `passwd` field contains the password.

```
<form method="post" action="/login" id="login_login-main"
  class="login-form login-formside">
  <input type="hidden" name="op" value="login-main" />
  <input name="user" placeholder="user name" type="text" />
  <input name="passwd" placeholder="user password" type="password" />
  <div id="remember-me">
    <input type="checkbox" name="rem" id="rem-login-main" />
    <label for="rem-loginmain">Remember me</label>
    <a class="recover-password" href="/password">Reset password</a>
  </div>
  <div class="submit">
    <button class="btn" type="submit">Connect</button>
  </div>
</form>
```

- Also check that the login process does not require javascript. In the example above, the action is a relative URL: `/login`. This is the destination of the `POST` login request. If it is absent, and the post URL is generated by a javascript method like `onSubmit='validateForm();'`, it is unlikely the fetcher can submit it.

## Write a Correct Success/Failure Condition

A good condition must return:

- Failure if and only if the fetch result shows that you are not authenticated.
- Success if and only if the fetch result shows that you have successfully authenticated.

Body text match condition	<ul style="list-style-type: none"> <li>- Finding a string like <code>Welcome \$USER</code> in the content of the page may be a good success criteria.</li> <li>- Finding a string like <code>Please authenticate to access this document</code> would be a good failure test.</li> </ul>
Redirection condition	<ul style="list-style-type: none"> <li>- The fetcher tests only if a page contains a redirection to another page and does not follow the redirection (a page and its redirection are two different documents).</li> </ul>

- Many sites redirect unauthenticated requests to a login page. Assuming the first redirection destination is `/login/login.html`, a good condition would be **Failure if redirection matches** `/login/login/html`.  
Matching only the presence of a redirection is usually a bad idea. Sites often contain lots of invisible redirections, even when authenticated.

**Warning:** Login procedures requiring javascript are not supported, except in the simple case of auto-submitted forms (usually used to transfer cookies or session IDs across domains without setting large query parameters).

## Common Misconceptions

- Authentication process and authentication test (the Success/Failure condition) are two independent processes. The authentication test condition is tested every time a document is fetched, but the authentication itself is executed only when the condition returns a failure.
- The crawler and the fetcher are independent. The crawler does not know about authentication. The process is transparent. The fetcher is like a black box, accepting URLs as input and returning authenticated documents. That means that login pages must never be indexed. If it happens, it means that your authentication process is wrong: either the authentication process has failed and the fetcher is not authenticated, or the condition always returns success erroneously (and never triggers the authentication).
- Cookies are automatically enabled and handled, there is no need to enable them manually. Use the **Enable cookies for all sites** option only if an unauthenticated site requires them.  
  
Cookies are handled transparently by the fetcher. It behaves as much as possible like a standard web browser.

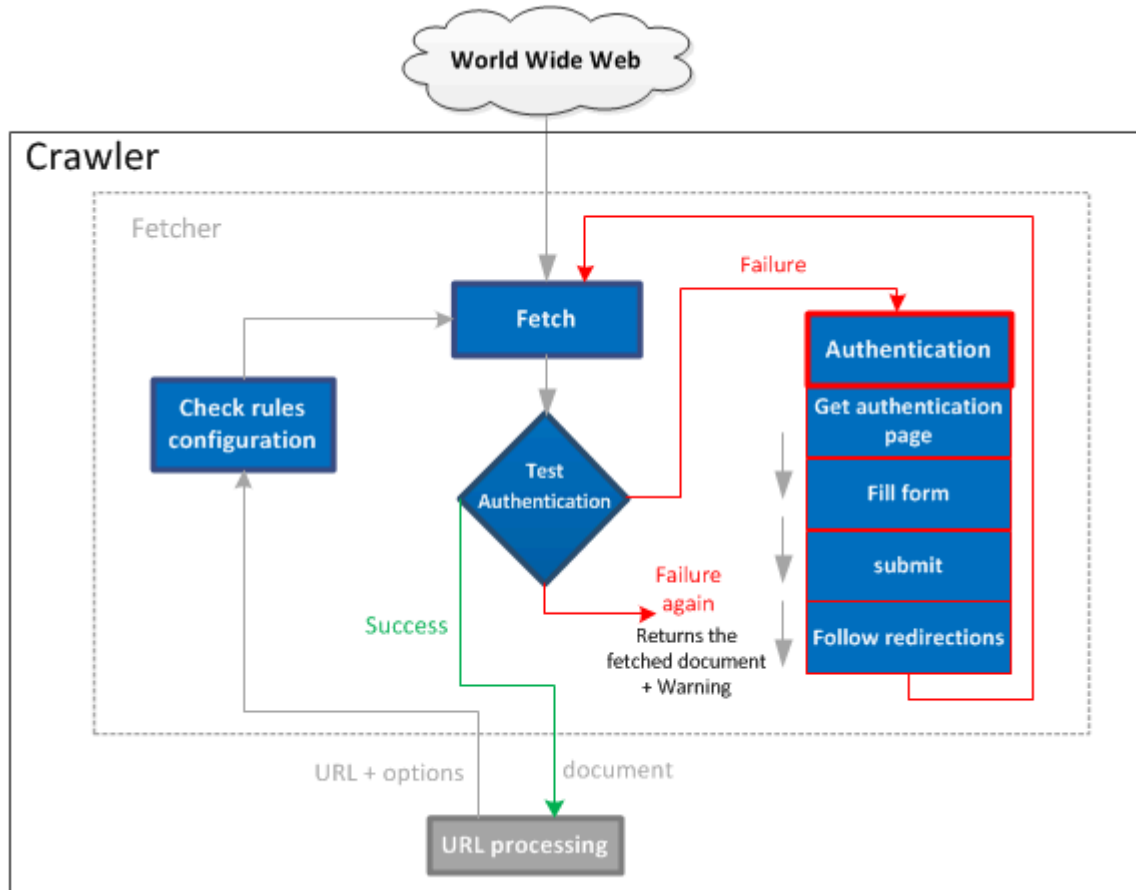
## Fetcher Authentication Process

This task shows you how the fetcher performs authentication.

1. The crawler asks the fetcher for a URL. The crawler does not know anything about authentication, and the fetcher does not know anything about the URLs that must be crawled and crawl rules.
2. The fetcher tests whether the URL matches the authentication configuration pattern.
3. The fetcher tries to fetch the URL a first time, without trying the authentication process. If the session has already been opened and is still active, a reauthentication is not required.
4. The fetcher tests the success/failure condition on the fetched document. If the condition returns "success", this means that the page is correctly authenticated, and is returned as is to the crawler.

5. Else, it means that the session has expired and the fetcher has to reauthenticate:
  - First, it fetches the gateway as configured. It follows redirections as required, until it finds an HTML page.
  - The login form is extracted from the HTML according to the configuration.
  - The parameters are extracted from the form, and completed with the username/password fields configured.
  - The form action is submitted with all these parameters.
  - Redirections are followed, with a limit to avoid loops. Redirections are often used at this point to transfer authentication tokens to session cookies (which can live on other domains).
6. Finally the fetcher tries to fetch the original URL a second time.
7. The success/failure condition is tested a second time: if the configuration is correct, it returns "success".
8. If the condition is successful, the document is returned to the crawler.
9. Else, the configuration is broken. A warning is printed in the crawler process log, and the resulting document is still returned to the crawler.

#### Fetcher Authentication Process





## Define the Groups of URLs to Crawl

## Define Crawl Rules

## Specify the File Name Extension and MIME Types to Crawl

## Sample Configurations for Different Use Cases

## Configuring the Crawler

This section implies that the Crawler connector has already been added.

See [Creating a Standard Connector](#).

### *Crawler General Options*

Option	Description
<b>Smart refresh</b>	<p>When enabled, the crawler continuously goes through the list of already crawled URLs and adds them to the crawling queue so they are refreshed.</p> <p>The URLs that have been refreshed or crawled recently - that is to say that have not crossed the <b>Min age</b> threshold - are not added to the crawling queue immediately.</p> <p>The URLs added by the refresh loop have a lower priority than the new URLs discovered during the crawl, so the refresh process does not slow down the discovery of new pages. For more details, see <a href="#">How Priorities Work</a>.</p>
<b>Min Age / Max age</b>	<p>Documents older than <b>Min Age</b> and younger than <b>Max age</b> are automatically refreshed depending on their update frequency. Documents that change often will be refreshed faster than documents that have never changed.</p> <p>In other words, documents are automatically refreshed according to their update frequency, but never faster than <b>Min Age</b>, and never slower than <b>Max age</b>.</p> <p>Reasonable values for a small intranet site are min = 1 hour, max = 1 week. Reasonable values for a big web crawl are much higher, at minimum 1 day and maximum 6 months or 1 year.</p>
<b>No. of crawler threads</b>	<p>Specifies the number of crawler threads to run simultaneously.</p> <p>If the crawler overloads your network, specify fewer threads.</p>
<b>Web server throttle time (ms)</b>	Specifies the lapse of time between queries (number of milliseconds).

Option	Description
	<p>By default, the connector sends one query at a time to a given HTTP server. This avoids overloading the HTTP server.</p> <p><b>Note:</b> The throttle time is ignored if you select the <b>Aggressive</b> mode. However, this may generate a heavy load on Web servers. Use it only when crawling your own servers.</p>
<b>Ignore robots.txt</b>	<p>Ignores the robots.txt that may be defined on the site you want to crawl. Caution! By default, the crawler follows the rules defined in the robot.txt to crawl the HTTP server.</p>
<b>Detect near duplicates</b>	<p>Detects similar documents.</p> <p>When a document is detected as identical to a previously crawled page (when the content is the same), it is not indexed and links are not followed.</p> <p>When it is only similar to a previously crawled document, it is indexed but links are not followed.</p> <p>This can avoid looping on infinitely growing URLs with duplicate content, and speeds up the crawl on sites where content can be accessed from different URLs.</p>
<b>Detect patterns</b>	<p>Detects sites based on specific patterns, such as CMS, parking site, SPAM, and adult content. It then applies the specific crawl rules defined for these sites. For example, it can detect search or login pages on forums that the connector must not crawl.</p>
<b>Convert processor</b>	<p>Searches for links in binary documents.</p> <p>To save CPU and memory, disable this option when crawling sites where there are no new links in binary documents.</p>
<b>Site collapsing</b>	<p>Activates site collapsing.</p> <p>By default, the site collapsing ID is based on the host part of the URL. All URLs belonging to the same host share the same ID and are shown as one collapsed result. You can configure the site ID to include more segments of the URL's path, to distinguish sites on the same host. This can be done by specifying the recursion with the <b>Depth</b> parameter. By default, the value is 0 corresponds to the host part of the URL.</p> <p>For more details, see <a href="#">About Site Collapsing</a>.</p>

Option	Description
<b>Default behavior</b>	<p>Allows you to specify a combination of index and follow rules for URLs that do not have any rules.</p> <ul style="list-style-type: none"> <li>• <b>Accept</b> – crawls URLs that do not match any rules. Beware, it can crawl the entire internet.</li> <li>• <b>Index</b> – indexes the contents of the documents at this URL.</li> <li>• <b>Follow</b> – follows the hyperlinks found in the documents at this URL. This finds content outside of this URL. Beware, it can crawl the entire internet.</li> <li>• <b>Follow roots</b> – follows root URLs that do not match any rules.</li> </ul>
<b>Default data model class</b>	Allows you to specify the class in which the documents must be indexed for URLs that do not have any data model class rules.
<b>License document type</b>	Select the type of documents that this connector produces.
<b>Blacklisted session ID</b>	<p>Allows you to specify the session IDs to be excluded (blacklisted) from the matching URLs.</p> <p>Many sites use session IDs which add parameters to the URLs randomly. This is troublesome since they can create different links for the same pages.</p> <p>For example, if you want to blacklist parameter 's' from URLs, enter <code>s</code> in the field. Consequently, if it finds a link to:</p> <p><code>http://foo.com/toto?s=123456789&amp;t=2</code></p> <p>... it crawls:</p> <p><code>http://foo.com/toto?t=2</code></p>

## Define the Groups of URLs to Crawl

The Crawler connector can have one or more groups containing root URLs.

Adding groups with **Add group** is a good means of organizing your sources by topic. For example, you can create a `news` group, to gather the source URLs of online newspapers.

1. In the **Groups** section, click **Add group** to add a new group.
2. In the **URLs** pane, enter the URLs of the sites OR the parts of the sites you want to crawl.

<p>Select <b>Site</b> to crawl the entire site. If you crawl an entire site, there is no need to define crawl rules.</p>	<p>- if the url is <code>http://foo.com/bar</code> it crawls only pages whose urls begin with <code>http://foo.com/bar</code></p>
--	---

	<p>- if the url is <code>http://foo.com</code> it crawls only urls beginning with <code>http://foo.com/</code> OR <code>http://www.foo.com/</code></p> <p><b>Note:</b> If the home page is a redirection or a meta redirection to another site, it follows it and crawls the destination site. Except for the home page, it does not follow redirections to pages external to the site.</p>
If you want to restrict the crawl, clear the <b>Site</b> option and define crawl rules.	<p><b>Note:</b> If you have used the <b>Site</b> option and then want to use advanced rules, clear the crawler documents first, otherwise unexpected behavior may occur.</p>

- If you define several source URLs, you can sort their crawling priority from the **Priority** select box. See [How Priorities Work](#).
- Add more groups and define the URLs to crawl.

## Define Crawl Rules

If you do not want to crawl an entire site, you can define precisely how to crawl each root URL in the HTTP source, using crawl rules.

These rules allow you to specify the action to perform for each url. For example, follow the hyperlinks and index the contents found on the pages.

Remind that the crawler applies the rule to all URLs in all groups. The pattern is a prefix matching the URL's beginning. It must only match the URLs of its group, otherwise unexpected behavior may occur.

- Expand **Advanced rules**.
- Add as many rules as required and for each:
  - Specify a URL pattern.
  - Define the action to take when crawling URLs that match this specific pattern.

Action	Description
<b>Index and follow</b>	Indexes the contents at this URL. The links found in the page are followed and crawled.
<b>Index and don't follow</b>	Indexes the contents but ignore the hyperlinks found in the page.
<b>Follow but don't index</b>	Follows the hyperlinks found in the pages at this URL but do not index the content.



Action	Description
<b>Index</b>	Indexes the contents of the pages at this URL.
<b>Follow</b>	Follows the hyperlinks found in the pages at this URL. This finds content outside of this URL.
<b>Don't index</b>	Does not index the contents at this URL.
<b>Don't follow</b>	Ignores the links found in the page.
<b>Ignore</b>	Ignores the defined URL completely.
<b>Source</b>	For compatibility with version 5.1 where several sources could be defined for the same crawler.
<b>Add meta</b>	Adds a meta as a key/value pair to flag the contents and hyperlinks of the pages at this URL.
<b>Priority</b>	If you define several crawl rules, you can sort their priority from the <b>Priority</b> select box. This changes the priority of URLs matching the pattern. See <a href="#">How Priorities Work</a> .
<b>Data model class</b>	Allows you to specify the data model class of the documents pushed to the index.

For example, we can crawl a single URL “`http://www.example.com`” and define the following patterns and actions:

- `http://www.example.com/` – **index and follow**
- `http://www.example.com/test` – **ignore**
- `http://www.example.com/test/new` – **index and follow**
- `http://www.example.com/listings/` – **don't index**

**Note:** You can quickly check the effect of your crawl rules (whether they work or not) in the **Test rules** section. This is interesting when rules are complex and you want to make sure that they do not break anything before applying the configuration to the crawler. Select **Advanced mode** to specify the expected behavior for each URL, and test rules without applying the configuration. Check boxes turn to green when the actual behavior matches the expected one, otherwise, they turn red.

3. Use the up and down arrows on the right of the **Actions** field to sort the rules. Precedence is given to the last matching rule (the last rule has more priority).
4. Expert users can also click **Edit rules as xml** to fine-tune rules manually. See [Advanced Configuration](#).

## Specify the File Name Extension and MIME Types to Crawl

Exalead CloudView gives you the possibility of specifying the file name types that the crawler must take into account. This allows you to focus only on interesting files, and reduce the amount of information to crawl.

By file name types, we refer to:

- The file 'extensions' that you can filter before crawling URLs.
  - The 'mime types' that Exalead CloudView identifies once a document has been fetched.
1. In **Filename extension and MIME types**, select the extensions and mime types to include or exclude.
  2. Click **Apply**.

## Sample Configurations for Different Use Cases

### Crawl One Intranet Site

When crawling only intranet sites, the configuration is simple, you must enter root URLs and maybe some rules to avoid indexing useless documents.

The server is friendly, so you can remove crawl speed limits. In that case, we say that you can configure the crawler in "aggressive" mode, which means that there is no throttle time between requests. This allows faster crawl and refresh.

You can also enable "smart refresh" to index new documents quickly. This option refreshes pages that change more often automatically, though it needs some adaptation time.

If required, you can configure a deterministic refresh, to ensure that all pages refresh in a given period. You have to check that the refresh period is long enough to recrawl all pages, using the server response time. You can schedule deterministic refresh using the product's scheduler. See [Refresh](#).

### Crawl an Intranet Site with SSO Authentication

The crawler configuration is the same as above, but the fetcher needs some authentication configuration too.

Follow the instructions. Usually you can copy-paste the URL patterns from the crawl rules for the authentication configuration.

### Crawl Many Intranet Sites

It is better to follow these recommendations when crawling many different Intranet sites.

- Usually, one crawler containing all the sites is more efficient than many crawlers. Indeed, the crawler can share resources (CPU, memory, threads), whereas two independent crawlers consume twice the resources without being faster.
- The only good reasons to separate the sites in several crawlers are the following:
  - You need different crawler options (throttle time, aggressive mode, refresh).
  - You need documents indexed in a different build group.
  - You deploy different crawlers on different hosts for performance.
- Deterministic refresh is still an option, see [Crawl One Intranet Site](#).
- The number of threads is important. Having too many threads uses too much memory, and few are able to crawl fast enough. To find the appropriate number, remember that you can crawl a host with only one thread at a time, so threads above the total number of hosts are useless. You can also fine-tune using the performance graphs in the Monitoring Console: the crawler has a graph showing the average number of idle threads. If many threads are idle, you can lower their number.

## Crawl Specific Internet Sites

If you need to crawl specific pages on a small number of sites (for example, product pages on e-commerce sites, or articles on a news site), write crawl rules carefully to define precisely which URLs to index, and which URLs to follow but do not index.

The easiest way is to use the **Site** option. This way, the crawler crawls every URL in the site. With some additional rules, you can select which pages to index among the crawled documents:

- A rule with the same pattern as the whole site, and **don't index** action overwrites the default for the **Site** option. Now the site is still crawled, but no longer indexed.
- Enough rules to match the pages you want to index with the **index** action. This overwrites the previous rule only for the wanted pages. You can also use the **index and don't follow** action if the links from these pages are not required.

To discover new links earlier without missing any, use **Smart refresh** or a specific deterministic refresh on intermediate listing pages. For more information about deterministic refresh, see [Refresh](#).

Be aware of the limited crawl speed (usually 1 document every 2.5 seconds) on the internet, which defines an upper bound on the number of crawled pages per day. You can tune the refresh periods with the **Min/Max** document age options. To optimize the crawl behavior:

- Increase the acceptable time to slow the refresh,
- Decrease it if you have available resources to enhance freshness.

## Crawl Many internet Sites

If you need to crawl a large or very large list of websites, configuring them in the Administration Console becomes a chore.

The easiest way to achieve this task involves several features:

- Site mode,
- Root sets.

The **Site** mode allows you to avoid setting crawl rules per site.

The **root set** configuration allows the crawler to refresh the list of sites to crawl dynamically, from a file outside the configuration.

The root set URL locates the file containing the site list. It can be a regular file:

- On the local filesystem (`file://`),
- In the product's data directory (`data://`),
- In the configuration directory (`config://latest/master/`),
- Or even a remote server (`http://`).

The file can be as simple as one URL per line.

You can also store the root set in the resource manager, using a `resourcemanager://` URL. For more information, see the resource manager's documentation.

Refresh works better with **Smart refresh**, and longer time periods (1 week to 1 month), to avoid unnecessary aggressive refresh.

The crawler can have as many crawl threads as the hardware resources allow. 20 to 100 threads are generally used in large crawls.

You can categorize sites using the root set group (split rootsets into several smaller sets, identified by their group), or on a per-site basis using forced metas. Example of root set using forced metas:

```
http://www.example1.com/ category=news theme=sports relevancy=100
```

```
http://www.example2.com/ category=blogs relevancy=10
```

To optimize resource usage, you can configure a maximum number of crawled pages per site. The default value is 1 million pages.

## Configuring the Fetcher

You can configure the fetcher of the Crawler connector in the **CRAWLER CONNECTOR > Fetcher** tab as described below. This subsection details how to configure the fetcher general

parameters as well as the rules for the fetcher headers (the HTTP headers that are sent to all URLs).

## Configure the Fetcher

1. In the **Fetcher** tab, expand **More options**.
2. Select **Enable cookies for all sites** to enable cookies for all sites, not only site authenticated with an HTML form and a cookie.
3. In **User agent**, enter the User-Agent header string to send in HTTP requests.
4. In **From**, enter the email address sent in the HTTP header.
5. In **Proxy address**, enter the proxy server address.
6. Define your proxy server credentials in **Proxy username** and **Proxy password**.
7. Define the timeout parameters:
  - **Read timeout (s)**: The read timeout expressed in seconds.
  - **Write timeout (s)**: The write timeout expressed in seconds.
  - **Connect timeout (s)**: The connect timeout expressed in seconds.
  - **Download timeout (s)**: The maximum connection time before canceling a download in seconds.
8. Click **Apply**.

## Configure the Fetcher Headers

1. Click **Add header** to add a new header.  
For all header rules, specify the header name and value.
2. In **Name**, enter the header name. For example, `Accept-Language`
3. In **Value**, enter the header value. For example, `en`
4. Click **Apply**.

## Crawl Secured Sites

This subsection describes the parameters that you can configure to crawl secured sites, from **Fetcher > Rules Configuration**.

Some configuration parameters can be rule-based. Each configuration whose rules match the URL to be fetched is applied.

Ensure that there are no conflicts in the definition of your Fetcher configuration rules. For example, do not define two different authentication schemes with overlapping rules. Each rule-based configuration depends on the type of rule.

## Choose an Authentication Type

From the **Auth type** select box, you can choose one of the following authentication types.

Auth type	requires...
<b>Basic</b>	Valid user name Clear text password
<b>Digest</b>	Valid user name Clear text password Realm
<b>Ntlm</b>	Valid user name Clear text password NTLM (Microsoft Windows NT LAN Manager) domain and host to authenticate users to Microsoft Windows 2000 domains.
<b>Form</b>	A login through an HTML authentication form with cookies handling.

## Configure Authentication Parameters

Once an Authentication type is selected, configure its related parameters. Below are example procedures for **basic** and **form** authentication type.

### Configure Basic Authentication

1. Click **Add rule**.
  - Enter a rule name, for example `user`.
  - For **Type**, select **Authentication**
  - Click **Accept**.
2. From **Auth type**, select **Basic**.
3. In **Username**, enter a valid user name for the authentication type.
4. In **Password**, enter a valid password for the user name.
5. In **URL patterns**, enter the site prefix to be authenticated. For example, `http://example.com/private/basic`

6. Click **Add pattern** to enter additional site prefixes using the current authentication configuration.
7. Click **Apply**.

The configurations of the **Digest** and **Ntlm** authentication types are similar. For **Digest**, also define the **realm** authentication. For **Ntlm**, also define the **domain** name and the **host** name.

### Configure Form Authentication

1. Click **Add rule**.
  - Enter a rule name, for example `user`.
  - For **Type**, select **Authentication**
  - Click **Accept**.
2. From **Auth type**, select **Form**.
3. In **Login page URL**, enter the URL of the page on which to find the authentication form.
4. If there are several html forms on the login page, specify either a CSS id or a class to identify which form must be submitted in **Form Id**, **Form Class** or **Form Name**. These parameters are not required if there is a single form.
5. From the **Form fields** pane, configure the fields required to authenticate on the authentication form, using the **Name** and **Value** fields. Specify the **input name** of the field, not its label.

**Note:** The best way to find input names is to open the source HTML code of the page and search for `input` nodes with attributes `type="text"` and `type="password"`, and copy the values of their `name="<input name>"` attributes.

Click **Add form field** for additional fields or **Add encrypted form field** if required.

For example, you can add:

- `username` – John
- `password` – \*\*\*\*\*
- `lang` – EN

**Note:** These fields are case-sensitive.

6. The Crawler Connector must know whether a fetched page is successfully authenticated or not. This is expressed with a condition based on **Authentication success** or **Authentication failure**. Pages that fail the test trigger a new authentication and are fetched again.

Select the **Success/Failure condition** based on authentication success or failure:

- if a redirection occurs

- if status is <value>
- if header with name <name> is present or equals <value>
- if body contains <value>

OR click **Edit condition as xml (experts only)** to enter a complex condition.

Examples:

- Set the condition to authentication failure if redirection URL matches "/login.php", that is to say, if the authentication fails and redirects you to the login page.
- For a private forum, you can set the condition to: authentication failure if body contains "Please login" OR authentication success if body contains "Welcome John".

**Note:** For further details about these conditions, see [About Success/Failure Conditions](#).

7. In **URL patterns**, enter the site prefixes to be authenticated. For example, `http://www.example.com/`.

**Note:** It might seem redundant with the URL specified in the **Configuration** tab but the fetcher needs to know for which URLs of the site, authentication is required.



Rules configuration
Add rule

user
Authentication

Auth type
Form

Login page URL
http://www.example.com/login.php

Form class

Form id
id\_login

Form name

Form fields

Name	Value	
username	John	X
password	*****	X
lang	EN	X

Add form field
Add encrypted form field

Success/Failure condition

Authentication failure
if redirection
URL matches
/login.php

Edit condition as XML (Experts only).

URL patterns

Pattern
http://www.example.com/

Add pattern

Edit as XML (Experts only)

8. Click **Add pattern** for additional site prefixes.
9. Click **Apply**.

### About Success/Failure Conditions

We must know whether a fetched page is successfully authenticated or not. This is expressed with a condition based on **authentication success** or **authentication failure**. Pages that fail the test, trigger a new authentication, and are then fetched again.

Condition	Description
<b>redirection</b>	<p>Redirections include HTTP status codes 301, 302, 303, 307.</p> <p>The connector can also test redirections to a URL containing a specific string.</p>
<b>status</b>	Checks the HTTP status code for a specific value.

Condition	Description
	For example, you can specify the <code>200</code> OK status to indicate that pages are considered as authenticated.
<b>header with name</b>	Checks for a specific header in the HTTP response. <b>is present</b> – checks whether the header name is present or not. <b>contains</b> – check whether the header name is present with a given value.
<b>body contains</b>	Looks for a specific string in the body of the HTTP response. Applies only to non-empty documents with <code>text/*</code> content types. You can test it with a string to find either on non-authenticated pages (for example, a login form or prompt) or on authenticated pages ( <code>"welcome \$username"</code> , logout form).

## Deploying the Crawler Connector

### Specify the Crawler Server

When you create a new connector, you must specify which Crawler Server hosts it. If a Crawler Server does not already exist, you are prompted to create one when adding the new Crawler connector.

If a Crawler Server already exists, any new Crawler connector is automatically associated with it.

### Deploy a New Crawler Server

1. In the Administration Console, select **Deployment > Roles** and click **Add roles**.
2. In the list of roles, select the **Crawler Server** and click **Accept**.
3. In the new **Crawler Server** box:
  - For **Instance name**, enter a unique name. For example, `exa1`.
4. Click **Apply**.

### Specify the Push API Server

Before a connector can push documents to Exalead CloudView, it must retrieve the URL of the Push API server with which it communicates. By default, this is **Build group bg0**.

### Change the Build Group for a Connector

1. On the Administration Console **Home** page, under **Connectors**, click the connector name.

2. Go to the **Deployment** tab.
3. For **Push to PAPI server**, select the appropriate Push API server from the list.
4. Click **Apply**.

## Managing the Crawler Connector

You can perform the following operations for the Crawler Connector.

From the **Connectors > CRAWLER CONNECTOR > Operations and Inspection** tab:

- **Stop/Start crawl** – stops or starts the crawl threads.
- **Clear documents** – deletes all connector documents from the system and the connector state is reset.
- **Clean documents** – deletes crawled documents in the index if they do not match the existing applied rules. It is useful if you have removed sites from the configuration, or applied stricter crawl rules, and want to get rid of "unwanted documents".
- **Reindex documents** – reindexes all documents from the crawler storage (it does not need to download them again).
- Expand **Advanced inspection** to monitor the scheduler state.

## Troubleshooting the Crawler Connector

This section describes how to monitor and troubleshoot the Crawler connector.

### Use the Crawler http.log

In the Administration Console, the Crawler connector's crawl logs are available in the **CRAWLER CONNECTOR > Logs** tab.

They log all the actions performed for the URLs crawled by the crawler, with their HTTP response status and configuration messages. Stacks are also printed when unexpected exceptions occur.

These logs are saved in:

```
<DATADIR>/run/crawler-<crawler-server>/<crawler-name>.http.log
```

For example, for the default crawler server **exa0**, the path is: `<DATADIR>/run/crawler-exa0/HTTP_Crawler.http.log`

*Display of Crawler Connector logs in the Administration Console*

```

ng107.prod.exalead.com - c:\default - crawler-exa0 - HTTP_crawler.http.log - Thu Dec 08 12:11:52 GMT+100 2011
last 199 lines

2011/12/08-01:37:15 OK 200 OK http://www.nycom-int.com/news/142/2/2/53/0/17
- [refresh rdate=2011/12/07-19:32:17] [default: accept,index,follow (site)]
[content-type=text/html] [mime=text/html] [size=51867B]
[fetchDuration=142ms] [lang=en] [mime.html.simhash=4723924673675856984]
[mime.html.nbToken=3852] [diffWithPrevDoc=0] [postedLinks=2]

2011/12/08-01:37:18 OK 200 OK http://www.nycom-int.com/news/142/2/2/55/0/17
- [refresh rdate=2011/12/07-23:33:05] [default: accept,index,follow (site)]
[content-type=text/html] [mime=text/html] [size=51867B]
[fetchDuration=140ms] [lang=en] [mime.html.simhash=4723924673675856984]
[mime.html.nbToken=3852] [diffWithPrevDoc=0] [postedLinks=2]

2011/12/08-01:37:21 OK 200 OK http://www.nycom-int.com/news/142/2/2/56/0/17
- [refresh rdate=2011/12/07-23:33:08] [default: accept,index,follow (site)]
[content-type=text/html] [mime=text/html] [size=51867B]
[fetchDuration=144ms] [lang=en] [mime.html.simhash=4723924673675856984]
[mime.html.nbToken=3852] [diffWithPrevDoc=0] [postedLinks=2]

2011/12/08-01:37:25 OK 200 OK http://www.nycom-int.com/news/142/2/2/57/0/17
- [refresh rdate=2011/12/07-19:32:27] [default: accept,index,follow (site)]
[content-type=text/html] [mime=text/html] [size=51867B]

```

The `http.log` file contains one line for each processed URL, with lots of information about the result of its processing. This uses a logger named `crawllog-<crawlername>` at the level `info`, any default log level above `info` disables the `http.log`.

Field	example
date	2013/01/11-14:32:10
doc_status	OK
http_code	200
http_message	OK
url	http://www.reddit.com/r/announcements/
referrer	http://www.reddit.com/
messages	[default: accept,index,follow (site)] [content-type=text/html] [content-length=13347] [mime=text/html] [size=91204B] [fetchDuration=1655ms] [langInMeta=en] [lang=en] [mime.html.simhash=8857254644444405862] [mime.html.nbToken=1138] [postedLinks=297]

More information about these log fields:

- `doc_status` possible values are:
  - REDIR IGNORED
  - TEMPORARY\_ERROR
  - PERMANENT\_ERROR
- `referrer` is – when unavailable

- `messages` are printed chronologically as follows:
  1. Preprocessor rules (default rules or `rule<name/preprocessor>:rules`); latter rules override former ones, `accept/ignore` decides whether document is fetched.
  2. If the document is not ignored, the fetch result contains `content-type`, `content-length` as returned by server, verified mime, real document size (`content-length` is size before eventual decompression), `fetchDuration` in milliseconds.
  3. If the document is fetched, the processor outputs: `language`, `simhash` (similarity hash).
  4. Postprocessor rules, `index/noindex` decide whether document must be indexed, keep track stores document in box only.
  5. `PostedLinks` indicate the number of followed links (`follow rules` decides whether to follow any link from current document)

You can also consult the process logs of the Crawler Server from the **Troubleshooting > Logs** menu, by selecting a crawler server (for example, **crawler-exa0**) from **Processes** and clicking **Add**. These logs are saved in the `<DATADIR>/run/crawler-<Crawler Server>/log.log` file.

## Dump the Crawler Repository

You can use the `cvdebug` command-line tool, which allows you to perform debugging operations, to dump the entire or a part of the crawler repository.

Go to the `<DATADIR>\bin` directory and run the following command:

```
cvdebug box dump crawler=<CRAWLER NAME> [prefix]
```

For:

`crawler` – enter the name of the crawler to dump

`[prefix]` – Only dump URLs beginning with this prefix.

The format is space-separated, with a header.

Field	Description
<code>url</code>	Document URL
<code>redirUrl</code>	URL of redirection if any
<code>docType</code>	Document type ( <code>HTML</code> , <code>redirect</code> , <code>ANY</code> if unknown)
<code>httpStatus</code>	HTTP status: <code>STATUS_OK</code> , <code>STATUS_PERMANENT_ERROR</code> , <code>STATUS_FORBIDDEN</code>
<code>language</code>	Document language in ISO 639-1 alpha 2 format

Field	Description
lastSeenChangeDate	The last time the document has been seen as changed by the crawler
refreshDate	The last time the document has been refreshed by the crawler
firstCrawlDate	The first time the document was crawled
size	Document size in bytes
normalizedContentType	Document mime type (text/html, flash etc.)
encoding	Document encoding (utf-8, iso-8859-1, etc.)
truncated	Has the document been truncated? It occurs when the document size is bigger than the limits defined in <DATADIR>/config/ FetchConfig.xml
siteRoot	Specifies whether the document is a site root
site	Specifies under which site root this page has been crawled
rootGroup	Specifies the group this document belongs to
rulesGroup	Specifies which crawl rules group apply to this document
source	Specifies the source meta value as defined by crawl rules for which the selected action is <b>Source</b> . See <a href="#">Crawl Rule Actions</a> .
dataModelClass	Specifies the <code>dataModelClass</code> meta value as defined by crawl rules or which the selected action is <b>Data model class</b> . See <a href="#">Crawl Rule Actions</a> .

## Performance Monitoring

In the **Operations & Inspection** tab, you see the crawler status as well as several counts on the number of indexed documents, crawled documents, remaining URLs to crawl, and many probes.

The **Number of documents in crawler storage**, is the number of crawled documents. Usually, this number is higher than the number of documents indexed by the crawler. Some documents may have not been indexed because they were empty or reported errors, or were redirection errors. The crawler storage keeps a copy of all crawled documents to know which pages must be updated.

**Note:** In the Monitoring Console, crawlers have many graphs showing their overall state and activity under **HOSTNAME > Services > Exalead > Connectors > CRAWLER NAME**. You can see all values through the probes.

## Hardware Sizing

To correctly size resources for a crawler, you must get the following input:

- The total number of sites and pages expected.
- The average page size. It is usually 30kB on the web, more on Intranet sites containing office documents.
- The crawl speed and refresh period expected (consider the throttle time, crawl speed limit).

This section explains how to:

### Determine the Hardware Requirements

The total number of pages and average size gives roughly the disk space requirement.

#### Disk Space

The crawler uses space to store crawled documents. Pages are compressed. For an internet crawl, the average compressed page size is about 5kB. For 1 million documents, this amounts to 5GB.

The crawler needs between 2-3 disk accesses (I/O) for each crawled document. This corresponds to 2 IO/s/thread, so 100 IO/s for 50 threads. For reference, this saturates a standard SATA disk. Fast crawlers require RAID arrays, SAS disks, or SSDs.

During compaction operations, the crawler storage may temporarily use 50% more disk space.

#### CPU

Nowadays CPU is not a bottleneck for the crawler. A single AMD Opteron 2346 HE core (2007 CPU) supports up to 40 crawl threads.

#### Network Bandwidth

Network bandwidth use is directly computed from average document size and crawl speed.

On the internet, an average of 30kB per document means about 200kbps of bandwidth per thread, or 20Mbps for 100 threads.

#### Memory

A crawler has a base memory consumption of 500MB to 1GB, plus 50-100MB per thread, depending on crawled documents.

Stored documents also use about 40B. Example: with 50 threads and 10 million crawled documents, a crawler uses  $500\text{MB} + 50 \times 50\text{MB} + 10000000 \times 40\text{B}$ , or about 3.5GB.

## Temporary Memory Use

During compaction operations, the crawler may temporarily use twice the size of the 80 biggest crawled documents.

For 30MB pdf documents, this amounts to 4.8GB.

## Determine the Crawl Speed

The number of pages you need to crawl per day (refresh speed) determines the crawl speed.

On a single server, divide by the response time added to the throttle time. On the internet, 0.5 to 1 page per second per thread is the average speed, so one crawl thread is usually enough to crawl 2 to 5 hosts in parallel.

Example: Crawl 100 different hosts with an average of 10000 pages on the internet. A single host cannot be crawled faster than one page every 2.5 seconds, so each host can be crawled in  $10000 \times 2.5\text{s} = 25000\text{s}$ , about 8 hours. If you need to refresh every page every day, it makes  $100 \times 10000 = 1$  million pages per day, or about 12 docs/s. With a conservative thread crawl speed of 0.5 doc/s/thread, it corresponds to 24 threads.

## Advanced Configuration

You can configure the Crawler connector using the Crawl Manager of the Exalead CloudView Management APIs. Therefore you use the `<DATADIR>/config/CrawlConfig.xml` file to perform advanced configuration of the Crawler connector.

You can edit this XML file from the Administration Console by clicking **Edit as XML (Experts only)** links.

## Crawl Rules

Crawl rules perform Boolean matching on a URL string therefore the rules operate on any part of a URL (scheme, host, path, query, fragment), and match strings or regular expressions.

You can configure the behavior of the match to define whether it is case-sensitive or left and right anchored. For example, a rule of kind 'Length' matches the length of a part of the URL against an integer range. Shortcuts are available as shown in the table below.

**Note:** For all parts of the url, the `val` attribute is a regular expression. However, you must escape the special characters of a regexp `^$. *+? [ ] ( ) { } |` with a `\`.



Rule category	Rule key	Example	Description and parameters
Boolean	And, Not, Or	<code>&lt;Or&gt;...&lt;/Or&gt;</code>	Performs Boolean matching for the nested rules defined. Commonly used for Boolean matching on URL strings.
generic	Atom	<p><b>Syntax:</b></p> <pre>&lt;Atom field="" kind="" norm="" value="" /&gt;</pre> <p><b>Example:</b></p> <pre>&lt;Atom field="path" kind="prefix" norm="none" value="/ watch" /&gt;</pre>	<p>This generic rule needs to define the type of match:</p> <p>The field defines the type of element to match. Possible values are <code>url scheme host path query</code></p> <p><code>kind</code> defines the part of the field to use for the match. It can have the values: <code>exact prefix suffix</code></p> <ul style="list-style-type: none"> <li><code>inside</code> where you specify a regexp and its anchoring in <code>val</code>.</li> <li><code>length</code> where you specify the length of a field (<code>[ :10]</code>, <code>[11:12]</code>, <code>[30:]</code>) in <code>val</code>.</li> </ul> <p><code>norm</code> impacts the normalization level. The default is a case insensitive match that corresponds to <code>non</code>. Possible values: <code>norm lower none</code>.</p> <p><code>value</code> is a regexp that must be matched with the links during crawl, based on the <code>field</code> and <code>kind</code> parameters.</p>
shortcut	Domain	<p>Exact match on domain. For example,</p> <pre>&lt;Domain val="foo.com" /&gt;</pre> <p><b>Matches</b> <code>http://foo.com/</code> <b>and</b> <code>http://bar.foo.com</code> <b>but not</b> <code>http://barfoo.com</code></p>	<p>This is a shortcut for this combination of rule:</p> <pre>&lt;Or&gt;&lt;Atom field="host" kind="suffix" value=".foo.com" /&gt;&lt;Atom field="host" kind="exact" value="foo.com" /&gt;&lt;/Or&gt;</pre>

Rule category	Rule key	Example	Description and parameters
	Path	<code>&lt;Path val="/cgi-bin" /&gt;</code>	This rule is a shortcut for atom path-prefix. It is a left anchored match on the path.
	Ext	<code>&lt;Ext val=".gif" /&gt;</code>	This rule is a shortcut for atom path-suffix. It is a right anchored match on the path.
	Host	<code>&lt;Host val="www.wikipedia.org" /&gt;</code>	Performs an exact match on host. This rule is a shortcut for atom host-exact.
	Url	<code>&lt;Url val="http://en.wikipedia.org/wiki" /&gt;</code>	This rule is a shortcut for atom url-exact.
	Scheme	<code>&lt;Scheme val="http" /&gt;</code>	This rule is a shortcut for atom scheme-exact. Possible values: http https
	Query	<code>&lt;Query val="q=foo" /&gt;</code>	This rule is a shortcut for atom query-exact. It performs an exact match on the query.
	InQuery	<code>&lt;InQuery val="q=foo" /&gt;</code>	This rule is a shortcut for atom query-inside. It performs a match on the query not anchored.
	Length	<code>&lt;Length field="path" val="[30:]" /&gt;</code> matches URLs with a path length $\geq 30$	This rule is a shortcut for atom field-length. It specifies the length of the URL path.

**Note:** All objects belong to the namespace: `exa:com.exalead.actionrules.v21`

## Crawl Rule Actions

The table below lists the various crawl rule actions that you can use with their corresponding XML tags.

For example, a crawl rule with an **index and follow** action is written as follows:

```
<Rules group="example" key="auto">
  <Rule>
```

```

    <ar:Atom litteral="true" value="http://www.example.com/"
    norm="none" kind="prefix" field="url"/>
    <Index/>
    <Follow/>
    <Accept/>
  </Rule>
</Rules>

```

Action	adds XML Tags...
<b>Index and follow</b>	<Index/> <Follow/> <Accept/>
<b>Index and don't follow</b>	<Index/> <NoFollow/> <Accept/>
<b>Follow but don't index</b>	<NoIndex/> <Follow/> <Accept/>
<b>Index</b>	<Index/> <Accept/>
<b>Follow</b>	<Follow/> <Accept/>
<b>Don't index</b>	<NoIndex/>
<b>Don't follow</b>	<NoFollow/>
<b>Ignore</b>	<NoIndex/> <NoFollow/> <Ignore/>
<b>Source</b>	<Source name=""/>
<b>Add meta</b>	<AddMeta value="" name=""/>
<b>Priority</b>	<Priority shift=""/> <b>Possible values:</b> -2 = Highest -1 = Higher

Action	adds XML Tags...
	0 = Normal
	1 = Lower
	2 = Lowest

## How Priorities Work

When **Smart refresh** is enabled, the crawler scheduler may contain up to 6 URL sources; 5 fifos and 1 refresh source. If **Smart refresh** is disabled, you can use the Exalead CloudView scheduler to refresh sources at a specific time; URLs are sent to the `fifo:index`.

URL source	Number	Content	Default weight (priority)
fifo: user	0	Only user-submitted root URLs with priority 0, and roots with default priority.	10000
fifo: redir	1	Targets of redirections.	2000
fifo: index	2	Documents that are indexed but whose links are not be followed.	1000
fifo: index_follow	3	Documents that are indexed and whose links are followed.	100
fifo: follow	4	Documents whose links are followed, but which are not indexed.	10
smart refresh source	5	Documents to refresh.	1

The crawler scheduler picks URLs in each URL source according to their weight. The higher the weight of a fifo, the more links are picked.

If you define a crawl rule with a priority action, the priority shift is ascending or descending depending on the value:

- -2 = Highest,
- -1 = Higher,
- 0 = Normal,
- 1 = Lower,
- 2 = Lowest

## Error Handling

Many errors can occur when crawling a URL.

These errors are split into the following categories:

- Permanent errors
  - HTTP 404 Not Found
  - HTTP 304 Not Modified when GET was not conditional (inconsistent server behavior)
  - Redirection to malformed URL
  - HTTP 5XX server errors
  - HTTP 4XX content errors
  - DNS permanent errors (host not found, etc.)
  - Other connection errors
- Temporary errors
  - Connection time out, connection reset by peer, connection refused
  - HTTP 503 error with Retry-After header
  - DNS temporary errors (no answer)

The error status is remembered for each URL. When a URL triggers a permanent error, if a document was indexed for that URL, a deletion order is immediately sent to the Push API.

Documents in error are refreshed like other documents:

- When a URL is refreshed and triggers too many temporary errors, if a document was indexed for that URL, a deletion order is sent to the Push API, and its status is removed too. It will not be crawled again unless the crawler comes across a new link to it.
- When a URL is refreshed and triggers permanent errors, the URL status is removed. It will not be crawled again unless the crawler comes across a new link to it.

## Custom Configuration

You can customize the Crawler connector to override the standard crawler configuration using the `<DATADIR>/config/CrawlConfig.xml` file.

You configure the crawler customization by adding a `CustomCrawlConfig` object in the `Crawler` object.

You can customize each crawler individually through the following customization method.

The Crawler connector is in Exascript language. For more information, contact Professional services.

**Note:** You can rewrite the entire crawler pipeline or only parts of it. The missing parts are taken from the default one. Therefore, you can override a single processor of the crawler pipeline.

# JDBC Database Connector

This section describes the functionality and configuration of the JDBC Database connector.

[Introducing the JDBC Database Connector](#)

[Installing JDBC Drivers](#)

[Installing Custom Code](#)

[Configuring the JDBC Database Connector](#)

## Introducing the JDBC Database Connector

The JDBC Database connector allows you to index the content of an SQL Database accessed through JDBC. It uses configurable queries to extract records. Records are then processed to build the documents to index.

The record processing is configurable and can be extended by custom code.

This chapter explains how to deploy and configure Exalead CloudView JDBC Database connectors. You must have working knowledge of:

- The operating system on which the Exalead CloudView server and the JDBC Database are installed.
- The JDBC Database source structure.
- The Administration Console.

## Prerequisites

This connector requires:

- A JDBC driver for the JDBC Database you want to index.
- A user account that has access rights on the JDBC Database Server.

## Workflow

Before you can configure the JDBC Database connector specific parameters, you must:

- Install the driver. See [Installing JDBC Drivers](#).
- Create the connector. See [Creating a Standard Connector](#).

You then need to perform the following configuration tasks in this order:

- [Connect to the JDBC Database](#)

- [Specify the Query Parameters](#)
- [Define the Fields to Crawl](#)

## Installing JDBC Drivers

To connect to the JDBC Database server, you must supply the appropriate JDBC driver. You can install multiple drivers to the same directory as follows.

1. Copy the driver's JARs to the `<DATADIR>/javabin` shared directory.

**Note:** You can copy several drivers at once.

2. Restart the connector server process (if it already exists).
  - In the Administration Console, go to **Home**.
  - In the **Processes** pane, click the restart icon of the **connectors-java0**.

**Note:** If you want to point to an external library, in common with other applications, you can also choose to configure its filepath in the `<DATADIR>/config/Deployment.xml` file. The benefit of this method is that any change brought to the library is taken into account automatically (you only have to restart Exalead CloudView).

## Installing Custom Code

This section describes how to install your custom code in Exalead CloudView to work with the JDBC connector.

**Warning:** This customization method is specific to the JDBC connector. The standard method to include custom components is to package them as plugins. For more information, see .

1. Copy the custom file to:

`<DATADIR>/javabin`

**Note:** If you want to point to an external library, in common with other applications, you can also choose to configure its filepath in the appropriate `ProcessInternalConfig` block of the `<DATADIR>/config/DeploymentInternal.xml` file, as follows:

```
<ProcessInternalConfig ...
  <args>
    <StringValue value="-classpath <CLASSPATH>">
    </StringValue>
  </args>
```



```
</ProcessInternalConfig>
```

The benefit of this method is that any change brought to the library is taken into account automatically. You only have to restart the impacted Exalead CloudView process.

2. Restart the connector server process (if it already exists).
  - a. In the Administration Console, go to **Home**.
  - b. In the **Processes** pane, click the restart icon of the **connectors-java0**.

## Configuring the JDBC Database Connector

To get started with the JDBC Database connector, you must first select a driver, set the connection parameters and test the connection. Once the connection is established, you can enter a valid SQL query that allows you to automatically retrieve the fields for selection. To do so, you need a thorough understanding of SQL databases.

This section implies that the connector has already been added. See [Creating a Standard Connector](#).

### Connect to the JDBC Database

#### Specify the Query Parameters

#### Define the Fields to Crawl

### Examples of JDBC Database Connector Configurations

### Connect to the JDBC Database

Exalead CloudView provides a direct connection parameter to connect to your JDBC Database sources called the **Connection string**. You can provide the JDBC Database name, server address, and account information if required.

For example, `jdbc:mysql://<HOSTNAME>:<BASEPORT>/DATABASE`

The following is a sample procedure for configuring a Native MySQL database.

1. In the **Connection Parameters** pane:
  - a. In the **Driver** field, select one of the drivers installed. For example, `com.mysql.jdbc.Driver`
  - b. In the **Connection string** field, enter the JDBC URL of the Database. The syntax varies depending on the driver selected, for example, `jdbc:mysql://<localhost>/northwind`
  - c. In the **User** field, enter the username of the Database account to use.
  - d. In the **Password** field, enter the password for the account.

**Note:** If the password for that account is changed, you must also change it in Exalead CloudView.

2. Click **Test connection**.

The JDBC Database connector automatically connects to the JDBC Database, and the message **Connection is ok** displays.

3. Click **Apply**.

4. After you verify the JDBC Database connection, specify the queries that select the table fields to be crawled in the **Query parameters** pane. See [Specify the Query Parameters](#).

## Specify the Query Parameters

Documents are built by selecting columns from ordered lists returned by synchronization queries.

To index your database, you can configure the connector to use:

- A full synchronization mode and index the document corpus entirely.
- Or one of the incremental synchronization modes to push only a fraction of your database. Use incremental modes when your table contains either:
  - A timestamp representing the time of insertion
  - A unique serial number generated with `AUTO_INCREMENT`

During incremental synchronization, the connector fetches the current timestamp or serial number of the column, and populates a variable in the incremental query to retrieve new rows from the database.

For more information, see "Implementing Synchronization" in the Exalead CloudView Connector Programmer's Guide

### Step 1: Select a Synchronization Mode

The JDBC Database connector uses synchronization modes based on different indexing methods. You must first select one of them.

Mode	Description
<b>Full synchronization</b>	Pushes every document contained in the database when a <b>Full Scan</b> occurs (see <a href="#">Controlling Connectors</a> ).  This mode does not perform incremental synchronization.
<b>Column-based incremental synchronization</b>	This incremental synchronization mode performs:

Mode	Description
	<ul style="list-style-type: none"> <li>a first full synchronization as described in the <b>Full synchronization</b> mode</li> <li>and calculates the maximum values contained in the <b>Checkpoint column</b>. Supported SQL types for the maximum calculation are <code>TINYINT</code>, <code>SMALLINT</code>, <code>INTEGER</code>, <code>BIGINT</code>, <code>TIMESTAMP</code>, and <code>DATE</code>.</li> </ul> <p>This mode performs subsequent synchronizations as follows:</p> <ul style="list-style-type: none"> <li>The <b>Incremental variable</b> is populated in the <b>Incremental query</b> with the maximum value and used for the synchronization.</li> <li>A new maximum value is calculated.</li> </ul>
<b>Query-based incremental synchronization</b>	This mode is similar to the <b>Column-based incremental</b> mode except that is based on a query instead of a column checkpoint. A query, run before each synchronization, calculates the maximum value.
<b>Trigger-based incremental synchronization</b>	You can configure additional queries to maintain tables that keep track of added, updated, and deleted rows. Use these queries to set up triggers and maintain additional tables.

For large databases (> 1.000.000 entries), when indexing is not automatically run after a completed scan (**Force indexing after scan** option cleared), plan enough time between the initial scan and the first incremental scan to make sure that checkpoints are computed properly. Alternatively, you can click **Force commit** (**Administration Console > Home > Indexing** section) to make sure that all documents have been indexed before running your first incremental scan.

## Step 2a: Define the Main Parameters

The other parameters for the database queries are based on the selected synchronization mode. The required and optional parameters of each mode are described in the following procedures.

**Important:** The JDBC connector does not insert quotes automatically when populating values in incremental and fetch queries. This means that you must surround variables by quotes for dates and strings.

A typical incremental query using an integer column to select new documents: `SELECT * FROM table WHERE id_column > $(id)` becomes, for example, `SELECT * FROM table WHERE id_column > 42`.

A typical incremental query using a stamp column to select new documents: `SELECT * FROM table where stamp_column > {ts '$(stamp)'}.` Once again, use quotes here.

A typical fetch query using a varchar column: `SELECT * FROM table where id_column = '$(id_column)'` becomes, for example, `SELECT * FROM table where id_column = 'FRA0001'`. If you omit quotes, the query would be: `SELECT * FROM table where id_column = FRA0001`, which is not a valid SQL query.

**Important:** For Query-based and column-based incremental synchronization, you can define an incremental variable to populate the incremental query:

- Either by a verbatim string substitution (default option),
- Or by a driver variable substitution, which allows you to apply the same SQL value on different RDBMS.

To edit this option, go to the **Advanced** tab and configure the **Checkpoint replacement mode** property as required.

For the Full Synchronization Mode

1. Specify an **Initial query** to synchronize the entire document corpus. For example: `SELECT * FROM table`
2. Optionally, specify an **All URIs query** or a **Fetch query**.

For the Query-Based Incremental Synchronization Mode

1. Specify an **Initial query** to synchronize the entire document corpus. For example: `SELECT * FROM table`
2. Specify a **Checkpoint query** to fetch the value of the checkpoint stored in the **Incremental variable**. For example: `SELECT MAX(stamp_column) FROM table` or `SELECT NOW()`
3. Specify an **Incremental variable** that is to say, the variable name to populate in the **Incremental query** (contains the value of the **Checkpoint query**). For example: `STAMP`
4. Specify an **Incremental query** to perform incremental updates. For example: `SELECT * FROM table WHERE stamp_column > '$(STAMP)'`
5. Optionally, specify an **All URIs query** and/or a **Fetch query**.

For the Column-Based Incremental Synchronization Mode

1. Specify an **Initial query** to synchronize the entire document corpus. For example: `SELECT * FROM table`

2. Specify a **Checkpoint column**, that is to say the name of the column stored in the **Incremental variable** used with column-based incremental synchronization. For example:  
`stamp_column`
3. Specify an **Incremental variable** that is to say, the variable name to populate in the **Incremental query** (contains the value of the **Checkpoint column**). For example: `STAMP`
4. Specify an **Incremental query** to perform incremental updates. For example: `SELECT * FROM table WHERE stamp_column > '$(STAMP)'`

The connector will populate `'$(STAMP)'` with the maximum time stamp found in the column and use the query. For example: `SELECT * FROM table WHERE stamp_column > '2011-11-25 11:25:00'`

5. Optionally, specify an **All URIs query** and/or a **Fetch query**.

#### For the Trigger-Based Incremental Synchronization Mode

1. Specify a **Checkpoint query** to fetch the value of the checkpoint used as **Incremental Start Variable**.
2. Specify an **Incremental Start Variable** that is to say, the variable name that will be used to identify the next scan run.
3. Specify an **Incremental Variable** that is to say, the variable name that will be used to identify the last scan run.
4. Specify an **Initial Query** to synchronize the entire document corpus. For example: `SELECT * FROM table`
5. Optionally, for **Initial**, specify:
  - In **Initial Pre Queries**: The queries to run before the first synchronization.
  - In **Initial Post Queries**: The queries to run after the first synchronization. For example:  
`CREATE TABLE eventTable(id INT, action VARCHAR(1)) CREATE TRIGGER after_test_insert AFTER INSERT ON testTable REFERENCING NEW AS NEW FOR EACH ROW INSERT INTO eventTable VALUES (NEW.id, 'N')`
6. Specify an **Incremental query** to perform incremental updates. For example: `SELECT * FROM table WHERE id IN (SELECT id FROM eventTable WHERE action = 'N')`
7. Optionally, for **Incremental**, specify:
  - In **Incremental Delete Queries**: The queries to run to compute documents that will be deleted during incremental synchronizations. For example: `SELECT * FROM eventTable WHERE action = 'D'`
  - In **Incremental Pre Queries**: The queries to run before incremental synchronizations.
  - In **Incremental Post Queries**: The queries to run after incremental synchronizations. For example: `DELETE FROM eventTable WHERE action = 'N'`

8. Optionally, specify an **All URIs query**.
  - In **All URIs pre queries**: The queries to run before the deletion process.
  - In **All URIs post queries**: The queries to run after the deletion process.
9. Optionally, specify a **Fetch** query.

## Step 2b: Define Optional SQL Queries

The following table describes the optional queries that can be defined for each synchronization mode.

Query	Description
<b>All URIs query</b>	<p>The query used to list all document URIs in the database and compare them with those in the index.</p> <p>You can use it to detect entries deleted from the database, and reflect these deletions in the index. Write it to go through the primary keys of all the tables crawled by Exalead CloudView.</p> <p>For example:</p> <pre>SELECT products.productid, attributes.langid FROM products, attributes WHERE products.productid = attributes.productid</pre> <p>... selects all the rows corresponding to the union of the two primary keys <code>productid</code> and <code>attrid</code>.</p> <p>This query allows Exalead CloudView to quickly mirror deleted rows from the database.</p> <p><b>Note:</b> The field names that make up must be the same as the names returned by the main query.</p> <p>This query does not perform any accumulation. When a record belonging to an accumulated document is removed from the database, the document is left unchanged in the index.</p>
<b>Fetch query</b>	<p>The query used to fetch documents for previews. The preview lists the selected columns of every row returned by the query.</p> <p>This query does not perform any accumulation.</p> <p>Mapping names prefixed by a dollar sign (\$) are populated with the corresponding values contained in the document URI.</p> <p>For example, given an indexed document with the URI: <code>customerID=550&amp;</code> and a <b>Fetch query</b>:</p>

Query	Description
	<pre>SELECT * FROM customers WHERE customerID = \$(customerID)</pre> <p>The connector processes the query as follows:</p> <pre>SELECT * FROM customers WHERE customerID = 550</pre> <p>and uses this query to fetch the required document.</p>

### Step 3: Define Advanced Parameters

In **More options**, you configure the following optional advanced parameters.

Field	Description
<b>Auto commit</b>	<p>In trigger-based synchronization, select the auto commit mode on connection:</p> <ul style="list-style-type: none"> <li><code>true</code>: the database makes the commit</li> <li><code>false</code>: the JDBC connector makes the commit</li> </ul>
<b>Read only</b>	<p>Specifies if the connection to the database is read-only.</p> <p>Options: <code>true</code> <code>false</code> <code>&lt;empty&gt;</code></p>
<b>Query timeout</b>	<p>Specifies the query execution timeout in seconds.</p> <p>Options: <code>&lt;int&gt;</code> <code>&lt;empty&gt;</code></p>
<b>Max. field size</b>	<p>Specifies the maximum number of bytes to return for character and binary column values in a result set.</p> <p>Options: <code>&lt;int&gt;</code> <code>&lt;empty&gt;</code></p>
<b>Max. rows in RAM</b>	<p>Specifies how many rows a result set can contain at a time. When exceeding the maximum number, rows are dropped and not loaded in the result set. Your JDBC driver may not support that.</p>
<b>Verbose</b>	<p>Enables the debug mode in logs for the connector only.</p>
<b>Fetcher class</b>	<p>Specifies a custom fetcher class (java code) to fetch documents. Enter the class name.</p> <p>Custom classes must be in a jar file located in the connector directory:</p> <pre>&lt;DATADIR&gt;/resource/all-arch/builtin_plugins_deployed/jdbc-java-connector</pre>

**Recommendation:** Use the following settings when the database contains many files or text parts or if you encounter OutOfMemory errors:

- **Auto commit:** false
- **Read only:** true
- **Max rows in RAM:** lower the default value (5000)
- **Result set type:** FORWARD\_ONLY
- **Result set concurrency:** READ\_ONLY

## Define the Fields to Crawl

To retrieve the fields you must first configure the connection parameters, test the connection and enter the queries that select the table fields as previously shown.

### About Accumulation

Accumulation is the aggregation of field or column values on multiple rows that represent a single document. The Column Processor associated with this column determines the accumulation behavior. Accumulation occurs on consecutive rows that have the same document URI.

### Retrieve and Configure the Fields to Crawl

1. Click **Retrieve fields** at the bottom of the **Query parameters** pane.  
This automatically populates the connector with the table fields based on the query. You can then configure the parameters and column processors for each field as follows.
2. In the **Fields selection** pane, select the **use this field** check boxes of the fields to be handled by the connector.
3. Choose the selected fields that must be a part of the document URI by enabling their **Use as primary key** check boxes.  
The set of fields or columns included in the document URI determines how rows are accumulated.  
  
**Note:** There must be at least one field set as the primary key, otherwise Exalead CloudView cannot crawl the JDBC Database.
4. In **Meta Name**, enter the meta name or part to be added to the document. For example, `id` is pushed as the meta name.
5. Specify whether the field must be in verbose mode. For example, enter `false` if you do not want the verbose mode.



- Click **Add column processor** to add a new processor to the field.

An arbitrary number of processors can be associated with each field. These processors are responsible of handling field or column values. See [Column Processors](#) below.

- Click **Apply** to apply changes to the configuration.

You are now ready to scan and index your documents. See also [Controlling Connectors](#).

## Retrieve BLOBs from a Database

Our JDBC SQLite driver is able to retrieve data coming from BLOBs (Binary Large Objects) in a database table. BLOBs are typically image files like .jpg stored in dedicated table columns. To retrieve BLOBs, map them to the master part.

- In the **Fetch query** field, enter a query like the following to fetch data from the BLOB column:

```
select blob_id, blob_img from blobtable where blob_id=$(blob_id)
```

- In the **Fields selection** pane, expand the field in which you want to store BLOBs.
- Click **Add column processor**.
  - Select **Multiple Parts**.
  - Click **Accept**.
- In the **Multiple Parts** column processor configuration, for **Part Name**, enter `master`.
- Click **Apply** to apply changes to the configuration.

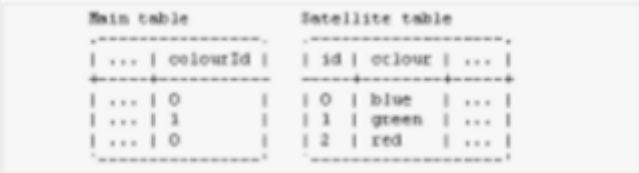
You are now ready to scan and index your documents with BLOB files. See also [Controlling Connectors](#).

## Column Processors

The column processors are described in the table below. All standard processors also accept the **Verbose** mode parameter.

Processor	Description
<b>Automatic</b>	Acts either like the <b>Multiple Parts</b> processor on BINARY/BLOB values or like the <b>Multiple Metas</b> on all other columns. If a meta is pushed, then its name is the value of the <b>Meta Name</b> setting; if a part is pushed, its name is <b>master</b> .
<b>Average Meta</b> <b>Max Meta</b> <b>Min Meta</b> <b>Total Meta</b>	Calculates the average/maximum/minimum/total value of the column and push the result as a meta with the name <b>Meta Name</b> .  These processors accept the <b>Multiplier</b> parameter. The result is multiplied by the <b>Multiplier</b> . This allows CloudView to push double values (double or floating) as integers.

Processor	Description
<b>Concatenate as meta</b>	Concatenates every value of the column and pushes the resulting string as the <b>Meta Name</b> meta. It accepts the <b>separator</b> parameter. The separator value is inserted between the different row values.
<b>Concatenate as part</b>	Concatenates every value of the column and pushes the resulting string as a part with the name <b>Part Name</b> . It accepts the <b>separator</b> parameter. The separator value is inserted between the different row values.
<b>Document Filter</b>	<p>Ignores or deletes the current document. It accepts the <b>Ignore Value</b> and <b>Delete Value</b> parameters.</p> <p>When the value of the column equals the <b>Ignore Value</b> (IGNORE by default), the resulting PAPI document is not pushed.</p> <p>When the value of the column equals the <b>Delete Value</b> (DELETE by default), the resulting PAPI document is deleted from the index.</p>
<b>File Attach Part</b>	<p>Pushes every value of the column as a part with the name <b>Part Name</b>. It accepts the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>Encoding</b> - encoding hint added to the resulting part.</li> <li>• <b>Encoding Column</b> - column containing encoding hints added to the resulting part (overrides encoding).</li> <li>• <b>Prefix</b> - prepended to the file name when attempting to load it.</li> <li>• <b>Suffix</b> - appended to the file name when attempting to load it.</li> <li>• <b>Mime</b> - mime hint added to the resulting part.</li> <li>• <b>Mime Column</b> - column containing mime hints added to the resulting part (overrides mime).</li> </ul>
<b>First Value as Meta</b> <b>Last Value as Meta</b>	Pushes only the first/last value (respectively) of the column as a meta with the name <b>Meta Name</b> .
<b>First Value as Part</b> <b>Last Value as Part</b>	Pushes only the first/last value (respectively) of the column as a part with the name <b>Part Name</b> .
<b>Custom</b>	Custom code processes every value of the column. The <b>Class Id</b> parameter is the java class of the column processor. You can enter additional parameters.
<b>Map Value as Meta</b>	Maps a column with a column found in another database (called satellite database). The mapped values are then pushed as metas with the name <b>Meta Name</b> . This processor accepts the following parameters:

Processor	Description
	<ul style="list-style-type: none"> <li>• <b>Class Name</b> Class of the satellite database driver.</li> <li>• <b>Connection String</b> - connection string used to connect to the satellite database.</li> <li>• <b>Query</b> - query used to list the values of the satellite table. This query produces results containing exactly two columns. The first column contains values to be populated. The second column contains replacement values.</li> <li>• <b>Login</b> - login used when connecting to the satellite database.</li> <li>• <b>Password</b> - password used when connecting to the satellite database.</li> </ul> <p>Example:</p>  <p>By attaching a <b>Map Value as Meta</b> processor to the column <code>colourId</code>, and setting the satellite query to:</p> <pre>SELECT id, colour FROM satelliteTable</pre> <p>This allows you to populate color ids with color names.</p>
<b>Multiple Metas</b>	With this processor, every value of the column is pushed as a separate meta value with the name <b>Meta Name</b> .
<b>Multiple Parts</b>	<p>With this processor, every value of the column is pushed as a separate part. The first part is pushed with the name <b>Part Name</b>. Subsequent parts are numbered, for example, <b>Part Name</b> is <code>master</code>, subsequent parts are named <code>master_1</code>, <code>master_2</code>.</p> <p>This processor accepts the <b>Filename Column</b> parameter, which designates a column that contains file names to be associated with pushed parts.</p>
<b>Row Num URI</b>	<p>With this processor, an identifier (integer) is generated and mapped automatically for each row of the tables that do not have a primary key.</p> <p>The enumeration order must stay stable over different enumerations. If not, Document URIs may become different over time and updates are not reliable.</p>

Processor	Description
<b>Unique Metas</b>	With this processor, every unique value of the column is pushed as a separate meta with the name <b>Meta Name</b> . This is the same as <b>Multiple Metas</b> but duplicates are removed. Order of values is kept.

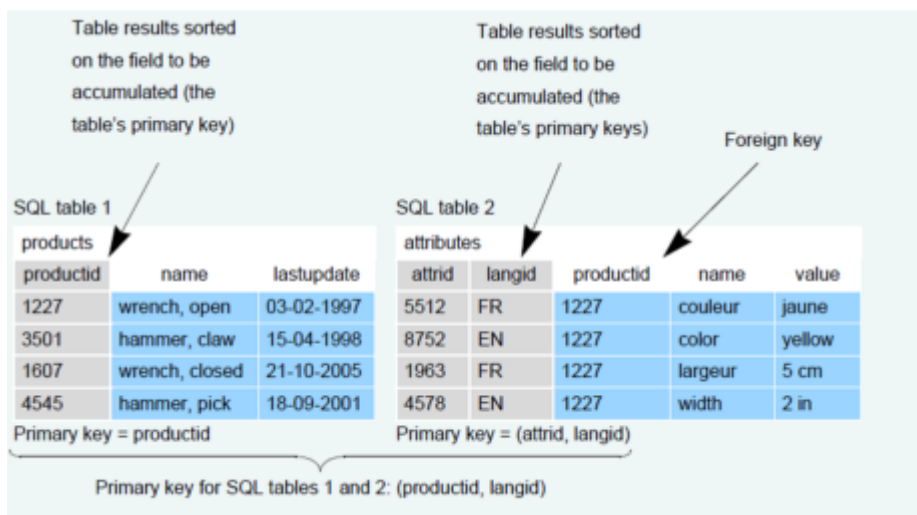
## Examples of JDBC Database Connector Configurations

This section gives examples of JDBC connector configurations.

### Index Cleanup

You can perform a query in **All URIs Query** to detect entries that were deleted from the database, so it can replicate these deletions in the index. Write this query to go through the primary keys of all the tables crawled by Exalead CloudView.

The following example shows a two table database to index with Exalead CloudView.



In SQL table 2, the `productid` field is a foreign key of the `productid` field stored in SQL table 1. The goal is to configure Exalead CloudView to index one product for each language `langid`.

For example, the SQL query:

```
SELECT products.productid, attributes.langid FROM products.attributes WHERE
products.productid = attributes.productid
```

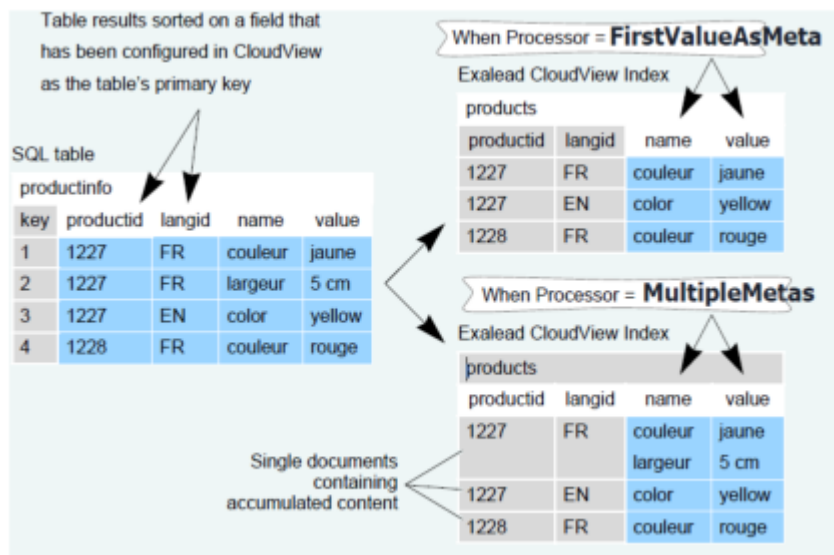
This selects all the rows corresponding to the union of the two primary keys `productid` and `attrid`. This query costs relatively little, and it allows CloudView to quickly mirror deleted rows from the database.

The field names that make up these keys must be the same as the names returned by the main query.

Accumulation is not performed in this step. When a record belonging to an accumulated document is removed from the database, the document is left unchanged in the index.

## Accumulation

The following example shows how the accumulation affects the indexing of a database table where `productinfo` contains many rows for each product.



Each row details a product specification that has been translated into many languages.

The goal is to accumulate all product specifications into a single document for each language in Exalead CloudView index. To do this, the **Initial query** must sort the results on the `productid` field and the `langid` field with the following SQL query:

```
SELECT * FROM productinfo ORDER BY productinfo.productid,productinfo.langid
```

**Note:** For accumulation to work, write the query to sort the database results on the fields defined as primary keys for this table.

Set the `productid` field and the `langid` field to be the primary keys for the table (as the primary key of the SQL table is of no search value). In this scenario, there may be more than two rows that have the same value for the primary key (`productid`, `langid`) to be indexed.

When you set the processor to **First Value as Meta** then only the first value for the `name` and `value` fields are kept.

When you set the processor to **Multiple Metas** then the `name` and `value` fields are merged for the rows.

## Configuring Trigger-Based Synchronization with Two Checkpoints

To make sure that scanning operations in progress do not miss any record, you can define two checkpoints for your trigger-based synchronization.

For this example, we use the database located in `<INSTALLDIR>/docs/sample_database/data.db`.

## 1. Customize the Database

We first customize the database to:

- Add a `product_order_log` table.
- Create 3 triggers for update, create, and delete operations.

Run the following SQL queries:

```
DROP TABLE IF EXISTS product_order_log;
CREATE TABLE product_order_log ("id" INTEGER PRIMARY KEY NOT NULL , "event_date"
DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP, "action" CHAR);
DROP TRIGGER IF EXISTS product_order_update_trigger;
DROP TRIGGER IF EXISTS product_order_create_trigger;
DROP TRIGGER IF EXISTS product_order_delete_trigger;
CREATE TRIGGER product_order_update_trigger UPDATE ON product_order
BEGIN
INSERT INTO product_order_log ("id", "action") VALUES (old.id, "U");
END;
CREATE TRIGGER product_order_create_trigger INSERT ON product_order
BEGIN
INSERT INTO product_order_log ("id", "action") VALUES (new.id, "C");
END;
CREATE TRIGGER product_order_delete_trigger DELETE ON product_order
BEGIN
INSERT INTO product_order_log ("id", "action") VALUES (old.id, "D");
END;
```

## 2. Configure Connection Parameters

Configure your connection parameters as follows:

- **Driver:** `org.sqlite.JDBC`
- **Connection string:** `jdbc:sqlite:<INSTALLDIR>/docs/sample_database/data.db`
- **User:** `<leave empty>`
- **Password:** `<leave empty>`

## 3. Configure Query Parameters

To make sure that all operations are taken into account, two checkpoints values are used:

- The last scan
- The next scan

Configure your query parameters as follows:

- **Synchronization mode: Trigger-based incremental synchronization**
- **Checkpoint query:** This query computes the checkpoint value based on the current date and time in the database.

Use the following query:

```
SELECT datetime('now') FROM product_order limit 1
```

- **Incremental variable:** This variable corresponds to the last checkpoint value (that is, computed during the last scan)

Enter: `previousCheckPoint`

- **Incremental start variable:** This variable corresponds to the next checkpoint value (that is, computed during the next scan)

Enter: `currentCheckPoint`

#### 4. Configure Initial Query Parameters

Configure your initial query parameters as follows:

- **Initial query:** Synchronize the entire table.

Use the following query:

```
SELECT id, date, quantity, customer_id, product_id, store_id FROM product_order
```

- **Initial post queries:** Delete triggers whose event date is before the initial scan run.

Use the following query:

```
DELETE FROM product_order_log WHERE event_date < '${currentCheckPoint}'
```

#### 5. Configure Incremental Query Parameters

Configure your incremental query parameters as follows:

- **Incremental delete queries:** Compute which documents need to be deleted.

Use the following query:

```
SELECT id FROM product_order_log WHERE event_date between '${previousCheckPoint}'
 '${currentCheckPoint}' AND action='D'
```

- **Incremental query:** Compute which documents have been created or updated since the last scan.

Use the following query:

```
SELECT p.id id, p.date date, p.quantity quantity, p.customer_id customer_id,
p.product_id product_id, p.store_id store_id FROM product_order as p,
product_order_log as l WHERE p.id=l.id AND l.event_date between
 '${previousCheckPoint}' AND '${currentCheckPoint}' AND l.action in ('C', 'U')
```

- **Incremental pre queries:** Delete triggers whose event date is before the last scan run.

Use the following query:

```
DELETE FROM product_order_log WHERE event_date < '${previousCheckPoint}'
```



# Feed Fetcher Connector

This section describes the functionality and configuration of the Feed Fetcher connector to crawl RSS, Atom, and RDF feeds.

## Introducing the Feed Fetcher Connector

This section implies that the connector has already been added.

See [Creating a Standard Connector](#).

The **General** configuration is the same as the Crawler connector's one. For more information, see [Configuring the Crawler](#). In the following use case, we want to crawl two feeds and refresh their crawls every hour.

## Configure the Connector

1. In the **General** pane, use the default options.
2. In the **URLs** pane, define the 2 RSS feeds to crawl.

Parameter	Setting
Feed	Enter the URLs of the feeds. For example, we can crawl the 2 following feeds: <code>http://www.example.com/feed1</code> and <code>http://www.example.com/feed2</code>
Refresh period(s)	Specify how often the feeds are refreshed. For example, 3600 seconds.
Actions	<p>Choose how to crawl the feeds:</p> <ul style="list-style-type: none"><li>• <b>Index feed and article content:</b> Indexes both what is displayed in the feed, and the content of the article (by following the link)</li><li>• <b>Index feed content only:</b> Indexes only what is displayed in the feed, NOT the content of the article.</li><li>• <b>Index metas quickly:</b> Indexes metas first when crawling the feed, and then again when crawling the entire article. This can be useful when you want to index article titles and summaries very quickly. The drawbacks are that your index will contain both partial and complete articles and you may have to configure your Mashup UI to avoid displaying empty hit contents (showing titles and summaries only).</li></ul>

Parameter	Setting
<b>Extract image/video links</b>	In addition to extracting media enclosures by default, look for links to images and links to youtube or dailymotion in the content of the feed items.
<b>Priority</b>	If you define several RSS feeds, you can sort their priority from the <b>Priority</b> select box. This changes the priority of URLs matching the pattern. See <a href="#">How Priorities Work</a> .

3. Click **Apply**.
4. On the **Home** page, click **Start crawl**.

The crawl of the 2 RSS feeds begins.

# Files Connector

This chapter describes how to configure the Files connector using Exalead CloudView Administration Console.

[File Server Access and Security](#)

[Mount Drive at Service Startup](#)

[About the Files Connector Configuration](#)

[Configure the Files Connector](#)

[Maximize Performances](#)

[Extending the Files Connector Through Plugins](#)

[Advanced Configuration Parameters](#)

## File Server Access and Security

You add a Files connector when you want to crawl:

- A local filesystem.
- A remote filesystem that is shared on the network.
- An HDFS file system.

The following topics are described below:

### Behavior

The behavior of the Files connector is dependent on the platform on which you install Exalead CloudView.

There are differences between UNIX and Windows platforms, as explained in [File Server Access and Security](#). In all cases, you can:

- Configure several paths to crawl.
- Exclude paths.
- Crawl based on file name extensions.

### Limitations

Consider the following limitations when implementing a Files connector:

- Windows servers have some limitations handling multiple connections to the same fileshare from different windows sessions. On Windows 2003 Server machines, this is configurable and depends on the Microsoft Windows license.
- Exalead CloudView cannot access remote files on a UNIX platform if they are mounted with the option `directio`. When using this option, the Files connector cannot index files within this filesystem. This is the case for CIFS (Common internet File System) mount points, as `directio` is defined as the default setting. Define the `nodirectio` option explicitly for that case. For more information, see the Linux man page for the UNIX command: `mount.cifs(8)`.

## Maximum Path Length

The maximum length for a path is defined as 260 characters for versions before Windows 10. You can override this limit and specify an extended-length path:

- Use the `\\?\` prefix for paths starting with a drive letter. Example: `\\?\D:\my_very_long_path`
- Use the `\\?\UNC\` prefix for paths starting with a server name. Example: `\\?\UNC\server\share`

## Local File Server Access

When you create a Files connector, you must ensure that Exalead CloudView has the required privileges to crawl the filesystem.

If the Files connector is installed on a UNIX server, the account privileges of Exalead CloudView are used to crawl the filesystem.

If the Files connector is installed on a Windows server, the account privileges of Exalead CloudView are also used to crawl the filesystem by default. However, Exalead CloudView often runs using the Local System user account, which has the same privileges as the administrator on local filesystems but no access to network filesystems. Therefore, if you want the Exalead CloudView user to have privileges to access network files, configure it to run with a domain user account during installation.

## Remote File Server Access

If the Files connector is not installed on the file server, it must access the connector's files remotely. Your network administrator must ensure that the filesystem to index is shared on the network and you have sufficient privileges.

## For MS Windows

If the Files connector is installed on a Windows platform, you can crawl a UNIX remote filesystem if it is shared on the network.

You can share filesystems on the network using SAMBA on the UNIX machine, export the corresponding share to the Windows machine, and then mount this share as a regular Windows share.

### Warning:

A CIFS support is provided by a third-party library. As it does not handle newer versions (only the SMB1 version of the protocol is supported), we do not recommend its use. Moreover, the next Exalead CloudView versions may no longer support it. The third-party CIFS handler is used if the URL starts with one the following conditions:

- `smb://`
- `\\` and an authentication is declared.
- `\\` and the OS is NOT MS Windows.

The Exalead CloudView `user` account is used to crawl the filesystem by default. Verify that this account has the required permissions to crawl the remote filesystem. The account must be valid for the domain.

If you need to define other authentication parameters, go to **Advanced > Root Paths > Item n > Connectivity configuration > Remote Windows filesystem advanced settings > Authentication** but remind that it only works reliably when DFS is involved (Distributed File System mount point).

### For UNIX

If the Files connector is installed on a UNIX server, you can access shared filesystems on the network using the system's own CIFS mount feature.

On Linux use the `mount.cifs` command or the corresponding `/etc/fstab` entry.

The Exalead CloudView `user` account is used to crawl the filesystem by default. Verify that this account has the required permissions to crawl the remote filesystem. The account must be valid for the domain.

### FTP Server Access

To crawl FTP or FTPS, define authentication parameters (username and password) in **Advanced > Root Paths > Item n > Connectivity configuration > FTP advanced settings > Authentication**. The URL format is the following: `ftp://server:<port>/<path>`

**Note:** SFTP is not supported.

## HDFS Server Access

To crawl HDFS, define authentication parameters (username and password) in **Advanced > Root Paths > Item n > Connectivity configuration > HDFS advanced settings > Authentication**.

The URL format is the following: `hdfs://server:<port>/<path>`

## Security

The default security profile for the Files connector allows users to only see the files that they would normally see, according to the filesystem's Access Control List.

Exalead CloudView indexes the ACLs for local users and groups with the files. The security tokens that are generated depend on the filesystem platform, for example:

- windows:S-1-5-21-34585....-5176
- UNIX:user:42
- UNIX:group:501

## Mount Drive at Service Startup

For easier access, you may need to mount a specific drive automatically each time the Exalead CloudView service starts.

1. Create a .BAT file specifying the drive to mount.

Example:

```
@echo off
net use /PERSISTENT:NO x: \\SERVER_NAME\MOUNT_POINT
rem we execute exit /b 0 so that the return code is 0 even if the net
use failed (e.g. already mounted drive)
rem you may have a more advanced error handling but the error code
will have no impact on CloudView.
exit /B 0
```

2. Edit the `DeploymentInternal.xml` file located in `<DATADIR>/config`.
3. To indicate the path to your .BAT file, add the following lines to `ProcessInternalConfig`

```
xmlns="exa:exa.bee.deploy.v10" name="connectors.*":
<StringValue xmlns="exa:exa.bee"
value="-Dcom.exalead.bee.Queen.preStartHookProcess=PATH_TO_BAT_FILE.bat"/>
```

**Note:** You can define `PATH_TO_BAT_FILE.bat` with placeholders like `${dataDir}`, `${runDir}`, `${configDir}`, or `${installDir}`.

## About the Files Connector Configuration

This section implies that the connector has already been added. See [Creating a Standard Connector](#).

### Filesystem Paths

You can configure the absolute paths that Exalead CloudView must crawl in **Filesystem paths**.

Directory names end by backward slashes (\) for MS Windows filesystems, and by forward slashes (/) for UNIX platforms.

**Note:** Do not specify the path to a single file, only paths to directories are supported.

### MS Windows Specificity for HDFS File System

In the `<DATADIR>/config/CloudviewDeploymentInternalConfig.xml` file, edit the `ProcessInternalConfig` node to add:

```
<StringValue xmlns="exa:exa.bee" value="-Dhadoop.home.dir=<path>"/>
```

where `<path>` is the path to a directory containing a `\bin\winutils.exe` file. The `winutils.exe` file does not have to be a valid executable file. Hadoop only checks its existence.

### Exclude/Include Rules

You can specify the path that you do not want to crawl by adding an **Exclude rule**. Similarly, you can specify the path that you want to crawl by adding an **Include rule**.

If there is	then...
No <b>Include rule</b> and no <b>Exclude rule</b>	all documents are accepted for the specified filename extensions.
One or more <b>Include rules</b>	documents are accepted if at least one include rule matches and if no exclude rule matches.
One or more <b>Exclude rules</b>	documents are accepted if no exclude rule matches.

**Note:** If the connector is running on Windows:

- Separate local paths with \, for example, `D:\temp`.
- Separate remote or network paths with /, for example, `//remote_host/temp`.

## Regular Expressions

You can use regular expressions (selecting **regex**) to specify the path.

The rule matches any substring within the absolute path of the file or folder. The folder path ends with a file separator (\ or /). If **regex** is selected, the expression has to be a valid regular expression. Otherwise, the expression is a substring.

For example, `C:\\Users\\*.\\Documents` matches the paths: `C:\Users\smith\Documents` and `C:\Users\dupont\Documents`.

## Examples

Root path is `C:\Documents` and you do not want to index `C:\Documents\Contracts\` except, `C:\Documents\Contracts\Public`

Possible solutions:

- Add two **Include** rules:
  - 1st rule is: `C:\\Documents\\Contracts\\$` with **regex** selected.
  - 2nd rule is: `C:\Documents\Contracts\Public` with **regex** NOT selected.
- Add one **Exclude** rule:
  - `C:\\Documents\Contracts\[^\$]+` with **regex** selected.

## Allowed Extensions

Specify which extensions to allow or ignore in the **Filename Extensions** section.

## Search Nonindexed File Names

By default, Exalead CloudView only crawls and indexes files specified in the **Filename extensions** list. This makes both their content and the filenames searchable. You can configure Exalead CloudView to search the filenames of nonindexed files.

For example, with **Index file names** set, users can search for executable files (`.exe` files) by their filename. Use this feature to find file types whose contents have little search value and that are not listed in **Filename extensions**.

## Configure the Files Connector

The procedures below show two types of Files connector configuration either through basic parameters in the **Configuration** tab or more specific parameters in the **Advanced** tab.



## Basic Configuration

The following procedure describes how to configure the Files connector basically, through the most common parameters available in the **Configuration** tab.

1. In **Filesystem paths**, enter the file system path to crawl.  
  
**Note:** This can be the mount point on the local host of a remote file system.
2. Click **Add path** to add more paths.
3. To exclude a subfolder from being crawled, expand **More options**, and enter the file system path of this subfolder in **Exclude rules**.  
  
Select **Regexp** if you want to use regular expressions in the path.
4. In **Filename extensions**, select what you want to crawl and index.
5. To make file names searchable for all documents that do not correspond to the **Filename extensions** list, select **Index file names for the ignored extensions**.
6. Click **Apply**.

## Advanced Configuration - Crawl HDFS Example

You can also configure the files connector partially or entirely in the **Advanced** tab.

In **Advanced > Root Paths > Item n > Connectivity configuration**, you can find several types of Files connector advanced settings for HDFS, FTP, HTTP, and Remote Windows filesystems. This section gives the example of a Files connector configuration to crawl an HDFS server.

1. Optionally:
  - a. In **File extension**, add or remove file extensions.
  - b. In **Max input size**, change the maximum size allowed for documents.
2. In **Root Paths > Item 0 > Root path**, enter the first file system path you want to crawl recursively. Use the format `hdfs://<HOST>:<PORT>/<PATH TO THE DIRECTORY>`.  
  
**Note:** This parameter is the same as the **Configuration** tab **>Filesystem paths** parameter.
3. Configure the HDFS server authentication if required.
  - a. Expand **Connectivity configuration**.
  - b. Click **Add item**
  - c. For **Component class name**, select **HDFS advanced settings**.
  - d. Expand **Authentication** and in **Username** and **Password**, specify the credentials of a user account allowed to crawl the HDFS filesystem.

4. To add more paths to crawl, go back to the **Root Paths** level, click **Add item** and repeat the two previous steps.
5. Click **Apply**.

## Maximize Performances

A default configuration (4 threads for folders, 4 threads for documents) is normally sufficient for most cases, the limiting factor being the filesystem/disks I/O. You can however consider tuning the following points to maximize the filesystem connector performances.

### Maximize Crawl Throughput

In the **Advanced** tab, you can change the value of the **No. pipeline document thread** and **No. pipeline folder thread** options.

Specify sufficient values for these options, to process more document/folder pipelines in parallel. For example, 32 for big configurations with a lot of I/O, or even higher if you can afford expensive RAID+SSD.

### Add a Metadata Compaction Preprocessor

With small files, when PushAPI performance is blocking, add a metadata compaction preprocessor to boost network I/O and remove contentions.

1. Go to **Advanced > PushAPI filters**
2. Click **Add item**.
3. In **PushAPI filter type**, select **Metadata Compaction PushAPI Filter**.

## Extending the Files Connector Through Plugins

The filesystem connector embeds natively a number of schemes and protocols: native filesystem, Windows share filesystem (`\\path` or `smb://` URLs), basic ftp support (`ftp://` URLs), basic http (`http://`), HDFS (`hdfs://`), etc.

It is possible to extend the features of the filesystem connector and use all the embedded features of the connector (multithreaded scan, containers handling, etc.) without having to create a new connector. You can do so by implementing additional protocol schemes through plugins.

Before going for a deep dive into customization, you must check whether:

- The source APIs are compatible with the `FileInterface` interface.
- The connector relies on hierarchical sources.

Therefore audit the source APIs to see whether you can crawl the source with a class implementing `FileInterface`. If it is OK, you can customize the connector. See the complete implementation procedure "Extending the Files Connector through Plugins" in the Exalead CloudView Connector Programmer's Guide.

## Advanced Configuration Parameters

This section describes the **Advanced** tab parameters.

Parameter	Description
<b>File extensions</b>	This is the text version of the <b>Configuration</b> tab <b>Filename extensions</b> section.
<b>Recursive</b>	Indexes sub-folders recursively. If unchecked, only the files in the defined top root paths will be indexed. Enabled by default.
<b>Enable ACL handling</b>	Fetches security tokens associated with files. <ul style="list-style-type: none"> <li>On Unix, it will fetch group/user security mode and, if available, POSIX ACLs.</li> <li>On Windows, it will fetch security SID.</li> </ul>
<b>Keep local ACL</b>	Only applies to Windows, and if <b>Enable ACL handling</b> is enabled. Fetches all security SID, including well-known local security SID such as "Local System"
<b>Skip directory symlinks</b>	Only applies to Unix/Linux. Skips symbolic links to directories (do not follow them) to avoid possible infinite loops.
<b>Default text encoding</b>	If specified, defines a global default encoding for text files on this connector. This encoding may be used to index raw text files whose encoding is unknown.
<b>Enable containers support</b>	If specified, files which are containers (i.e., ZIP files, TAR files, PST files, EML files, etc.) will be processed as if they were regular folders.
<b>Max. container depth</b>	When containers support is enabled, sets the maximum recursive depth inside containers. Example:

Parameter	Description
	<ul style="list-style-type: none"> <li>• A level of 1 will only allow file scanning within containers in the filesystem source.</li> <li>• A level of 2 will also allow to scan containers inside containers (a ZIP file in a ZIP file, for example) in the filesystem source.</li> <li>• A level of 3 will allow one further depth (for example, an attachment inside a mail inside a PST file).</li> </ul>
<b>Max. documents per container</b>	<p>When containers support is enabled, set the maximum number of files to be processed inside a single container (inside a ZIP file, for example).</p> <p>For example, considering the following structure:</p> <pre>foo.zip: a ZIP containing 80 files, and 10 ZIP files: file1.doc file2.doc ... file80.doc archive1.zip: a ZIP containing 50 files archive2.zip: a ZIP containing 50 files ... archive10.zip: a ZIP containing 50 files</pre> <p>Setting this value to "100" will allow to index all 80 files within <code>foo.zip</code>, and all 50 files within <code>archive1.zip</code>, all 50 files within <code>archive2.zip</code>, etc. The total number of files indexed will be equal to 580 (80 files at top level, and 50 files for each 10 archives).</p>
<b>Max. documents per container total</b>	<p>When containers support is enabled, set the maximum number of files to be processed overall, in all recursed container depth.</p> <p>In the previous example, setting this value to "100", will allow to index all 80 files within <code>foo.zip</code>, but the indexing will stop after indexing 20 files within <code>archive1.zip</code> file. Other archives will not be indexed at all.</p>
<b>CPath stop MIME filter</b>	<p>Define the MIME types of containers which are to be considered as documents as a whole. For example, msg or eml mail files are containers, because they may contain attachments or attached files themselves.</p>

Parameter	Description
	<b>Note:</b> If this parameter is empty, no restriction or exclusion is applied.
<b>Container MIME filter</b>	<p>Select the MIME types of files which are to be considered as containers.</p> <p><b>Note:</b> If this parameter is empty, no restriction or exclusion is applied.</p>
<b>Item MIME Filter</b>	<p>Select the MIME types of files to be scanned in a container.</p> <p><b>Note:</b> If this parameter is empty, no restriction or exclusion is applied.</p>
<b>Item extensions</b>	Define the extensions of files to be scanned in a container.
<b>Index names</b>	Push empty documents for all the files which have not been accepted because of filters. This allows to index filenames of files whose content should not be indexed.
<b>Max. input size</b>	<p>Maximum file input size allowed.</p> <p>Specify any SI byte unit (1000KB, 100MB, 1GB and so on). If no unit is specified, it uses bytes.</p>
<b>Max. container fetch size</b>	<p>Maximum container size allowed for fetch (preview, data fetch).</p> <p>Specify any SI byte unit (1000KB, 100MB, 1GB and so on). If no unit is specified, it uses bytes.</p>
<b>Convert address</b>	External Convert address. Should be empty to dispatch to default Converter.
<b>Container timeout</b>	When opening a container using a remote Convert service, define the timeout when opening the file. For example, a large PST file may take several minutes to be opened.
<b>Container fetch timeout</b>	When opening a container using a remote Convert service, define the timeout when fetching a sub-item.
<b>Truncate files pattern</b>	When a file is larger than the allowed size set in <b>Max. input size</b> , truncate the file rather than discarding it. This option is compatible only with raw text files, or HTML (not Office files or PDF, for example).

Parameter	Description
<b>Push folders as documents</b>	Push an empty document for all folders found. Disabled by default.
<b>Never send delete</b>	Never send any delete remotely, even if the file is no longer present locally. Disabled by default.
<b>Delete document on error</b>	<p>Define the strategy to be adopted when a document cannot be updated after a first indexing (if the file become unreadable, busy, or the access rights do not allow to access it anymore).</p> <ul style="list-style-type: none"> <li>• <b>Keep</b>: keep the entry in the remote index as it was before</li> <li>• <b>Delete</b>: remove the entry in the remote index</li> <li>• <b>Empty</b>: create an empty file in the remote index</li> </ul>
<b>Max. document queued</b>	Maximum number of documents to be added in the document processing queue (in memory).
<b>Max. folder queued</b>	Maximum number of folders to be added in the folder processing queue.
<b>No. pipeline document thread</b>	Number of background threads processing the document queue, that is, reading documents to be indexed and sending them to the remote server.
<b>No. pipeline folder thread</b>	Number of background threads processing the folder queue, that is, scanning locally folders to find all files and subfolders to be indexed.
<b>Max. processing size</b>	Limits the total amount of memory which can be used when processing the document queue. If the limit is reached, other document threads will be blocked until the memory is free.
<b>Root Paths (N)</b>	Text version of the <b>Configuration</b> tab <b>Filesystem paths</b>
<b>Filename include rules (N)</b>	Text version of the <b>Configuration</b> tab <b>Include rules</b>
<b>Filename exclude rules (N)</b>	Text version of the <b>Configuration</b> tab <b>Exclude rules</b>
<b>Main part MIME filters (N)</b>	<p>Used to aggregate and dedup items within a mail container. For example, this allows to index the HTML part of a mail, and ignore the text part.</p> <ul style="list-style-type: none"> <li>• <b>Parent MIME filter</b>: list of MIME filters of mail containers</li> <li>• <b>Main part MIME filter</b>: list of MIME types of body part(s) inside a mail</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Main part dedup MIME Fiter:</b> list of equivalent MIME types to be deduped</li> <li>• <b>Main part dedup max. count:</b> maximum number of documents to be deduped</li> <li>• <b>Add child links:</b> adds meta-data linking sub-child (such as attachments)</li> <li>• <b>Merge in parent:</b> merges bodies in the document</li> <li>• <b>Merge container metas:</b> merges container's metadata in main document</li> </ul>
<b>Filename MIME rules (N)</b>	<p>A set of rules allowing to set the MIME type, and optionally the encoding, of files matching the given extension/filename filter.</p> <ul style="list-style-type: none"> <li>• <b>Filter:</b> the space-separated list of filename extensions matching (or the regular expression, if the checkbox is checked)</li> <li>• <b>Regular expression:</b> if checked, the filter is a regular expression matching the filename</li> <li>• <b>MIME type:</b> the MIME type to set</li> <li>• <b>Encoding:</b> the encoding to set, optionally</li> <li>• <b>Hint only:</b> if checked, the MIME type is not forced</li> </ul>
<b>PushAPI filters (N)</b>	<p>The PushAPI pipeline configuration. Documents being added in the PushAPI pipeline will go through defined filters, starting by the first filter defined, until the last one, before being injected to the PushAPI.</p>

# IMAP Connector

This chapter describes how to configure the IMAP connector using Exalead CloudView Administration Console.

[About IMAP Configuration](#)

[Configure the IMAP Connector](#)

[Folder Configuration Examples](#)

[Troubleshooting](#)

## About IMAP Configuration

The IMAP connector indexes attachments and links them with the mail. All supported documents are indexed and the metadata is extracted.

You add an IMAP connector source for each IMAP server you want to crawl.

### Subject Filters

Subject filters are regular expressions that filter out individual emails from being indexed, based on words contained in the subject line. They work like anti-spam filters. These filters apply to all emails.

### Indexing Email Threads

A thread is a set of emails that are replies to one another. The IMAP connector lets you index documents that contain a complete thread. The text includes the message body of each mail in the thread, quotes being removed when possible to avoid redundancies.

For your information, the other metas pushed automatically by the connector include:

- `unanswered`: 0 if the mail has had an answer, 1 otherwise. The connector considers that a thread is answered if someone other than the original sender replied to the thread (A replying to A is considered as an unanswered thread).
- `rootsender`: author of the thread's first mail.
- `participants`: all the people that replied to the thread.
- `avgreplytime`: average time (in seconds) between two mails in this thread.
- `firstReplier`: first person to reply to the initial mail.



- `firstReplyTime`: time elapsed between the initial mail and its first reply in the thread (in seconds).
- `mailnumber`: number of mails in the thread.

## Gmail

The IMAP connector automatically detects when the IMAP server is a Gmail server.

In this case, the connector pushes additional metas for the following Gmail-specific variables:

- `X-GM-MSGID`: Gmail unique mail identifier
- `X-GM-THRID`: Gmail thread identifier
- `X-GM-LABELS`: Gmail labels

This allows you to access your emails directly in the Gmail interface by using the following URL:

`https://mail.google.com/mail/#inbox/X-GM-THRID`

## Subject Normalization

To detect when emails are part of the same thread, the basic algorithm uses the `References` or `In-Reply-To` fields. However, these can sometimes be corrupted, hence breaking the links. To overcome this issue, you can activate the **Subject normalization** option, so that threads and emails are grouped in one thread if they share the same subject (the analysis does not take the `Re :` strings into account).

The **Subject norm** is associated with the **Subject normalization** option. You can use it when email subjects share a common tag, for example `[Test]`, because some email clients will detect this tag and insert the reply tags (`Re :`, `R :`, etc.) after it. This makes the **Subject normalization** process inefficient. To ensure that the subject is normalized consistently, exclude this tag using the **Subject norm** field.

## Configure the IMAP Connector

This section implies that the connector has already been added.

See [Creating a Standard Connector](#).

## Configure the IMAP Server Global Configuration

1. In **Server name**, enter the name of the IMAP server.

For example, `imap.mycompany.com`

2. In **Server port**, enter the IMAP server's port number. It is usually 143 or 993 if SSL is activated.
3. Specify the following timeout limitations:
  - a. **Read/Write timeout** to specify the time limit in seconds for reading a message.
  - b. **Connect timeout** to specify the time limit in seconds for connecting to the IMAP server.
4. Select **SSL** if the server is a secured IMAPS server.
5. If **SSL** is enabled, you can select the **Authorize all certificates** property to allow users to access the IMAPS server without certificate authentication.
6. For efficiency, mails are retrieved by groups and loaded into memory. To avoid "out of memory" errors, it is therefore important to specify the two following limitations:
  - a. In **Max. fetched mails**, the maximum number of mails to fetch.
  - b. In **Max. fetched size**, the maximum size (in MB) allowed for a group of mails.

If one of these thresholds is crossed, the mail or group of mails, is not indexed and an error is displayed in the logs of the **connectors-java0** process. See [Troubleshooting](#).

## Configure a User Account

You can specify the users that have access to mail folders by associating them to a user account. You create user accounts under **User accounts** and you specify the security tokens associated to each user account in **Users**. For example, you can create the user account `marketing.team` and index the content of this account from `imap.gmail.com`. For this account, you define the login `marketing.team` and its associated password. You also create a list of users to restrain the access to the account to the users of this list, by listing their security tokens.

1. Go to **User accounts > Item 0**.
2. In **Login**, enter the login of the IMAP account, and in **Password**, enter the password associated to this login.
3. To index unread email messages, select **Index unread**.
4. To use the mail server's date instead of the date found in the message body, select **Use internal date**.
5. If your mail server has folders that you want everyone to have access to, select **Public**.
6. In **Users**, you can then specify a list of security tokens corresponding to the users allowed to access the indexed documents with this user account.

Copy the security tokens that have been defined under **Security Sources > Configuration**.

7. To secure the search access, you can then add a security source.
  - a. From **Search > Security Sources**, add a security source.
  - b. Create the users allowed to search for the documents of mail folders.

For more information on security sources, see "Managing User Access" in the Exalead CloudView Administration Guide

## Configure Folders

This procedure describes how to index mail folders and subfolders.

1. Go to **User account > Item 0 > Folders** and click **Add item**.
2. In **Name**, enter the name of the folder to index. Enter \* if you want to index all folders.
3. To include all subfolders in the indexing process, select **Recursive**.
4. If you do not want to index a specific folder, enter its name in the Name field and select **Ignore**.
5. To index thread objects, specify the following properties:
  - a. Select **Create threads** to index thread objects.
  - b. Select **Only threads** to index thread objects only (not individual mails).
6. To use subjects to group mails together in one thread object:
  - a. Select **Subject normalization**.
  - b. In addition, you can use the **Subject norm** field to exclude a string from the subject.  
See [Subject Normalization](#) .
7. In **Subject filters**, you can then specify a list of subject filters based on regular expressions. Mails with subjects matching these regular expressions are not indexed.
8. For example, the regular expression `.*noindex.*` matches the following mail subject lines:
  - `****NOINDEX****`
  - `Private: Salary survey. noindex`
  - `noindex Pension fund review`
9. Click **Save**.

## Check Connectivity

When you configure an IMAP connector source, it is good practice to ensure that Exalead CloudView can access the IMAP server.

1. At the top of the **IMAP connector > Configuration** tab, click **Check connectivity**.
2. If an error message appears, read the instructions to tackle the issue.

## Folder Configuration Examples

The following screenshots give configuration examples for indexing different folders on your IMAP server.

### *Configuration to Index All Folders Except the **Trash** Folder*

▼ Folders (2) ⓘ

▼ Item 0 ✕

Name	*	ⓘ
Recursive	<input checked="" type="checkbox"/>	ⓘ
Ignore	<input type="checkbox"/>	ⓘ
Create threads	<input type="checkbox"/>	ⓘ
Only threads	<input type="checkbox"/>	ⓘ
Subject normalization	<input type="checkbox"/>	ⓘ
Subject norm		ⓘ
▶ Subject filters (0) ⓘ		

▼ Item 1 ✕

Name	Trash	ⓘ
Recursive	<input type="checkbox"/>	ⓘ
Ignore	<input checked="" type="checkbox"/>	ⓘ
Create threads	<input type="checkbox"/>	ⓘ
Only threads	<input type="checkbox"/>	ⓘ
Subject normalization	<input type="checkbox"/>	ⓘ
Subject norm		ⓘ

### *Configuration to Index the **Private** Subfolder Contained in the **MyMails** Folder*

▼ Folders (1) ⓘ

▼ Item 0 ✕

Name	MyMails.Private	ⓘ
Recursive	<input checked="" type="checkbox"/>	ⓘ
Ignore	<input type="checkbox"/>	ⓘ
Create threads	<input type="checkbox"/>	ⓘ
Only threads	<input type="checkbox"/>	ⓘ
Subject normalization	<input type="checkbox"/>	ⓘ
Subject norm		ⓘ

## Troubleshooting

### Out of Memory Errors

The connector fetches multiple mails in one connection to the IMAP server for efficiency, instead of using one connection per mail, which would be much slower.

These mails are loaded in the memory of your JVM (Java Virtual Machine), which can lead to out-of-memory errors. If this problem occurs:

- Lower the **Max. fetched size** property, all the while increasing the **Max. fetched mails** to avoid losing too much performance. However, if an email's size is bigger than the **Max. fetched size**, it is not indexed and a warning is issued in the connector log.
- If possible, increase the memory allocated to the JVM.

### Mails Do Not Group by Thread Subjects

Depending on the mail server, it happens that emails from a same thread are not grouped all together.

To tackle this issue, you can activate the **Subject normalization** option to group mails with the same subject. For more information, see [Subject Normalization](#) .

# LDAP Connector

This chapter describes how to configure the LDAP connector using Exalead CloudView Administration Console.

[About LDAP Configuration](#)

[Configure the LDAP Connector](#)

[Parameter Descriptions](#)

## About LDAP Configuration

### Before You Start

- Before you configure your LDAP connector specific parameters, you must create your connector as described in [Creating a Standard Connector](#).

**Note:** Add one LDAP connector for each LDAP server to crawl.

- When you configure an LDAP connector, make sure ensure that Exalead CloudView can crawl your server.
- Decide how you want Exalead CloudView to crawl your LDAP connector. See [LDAP Classes and Attributes](#).
- There is no inherent security in LDAP. If you want to implement security, you must create a custom connector.

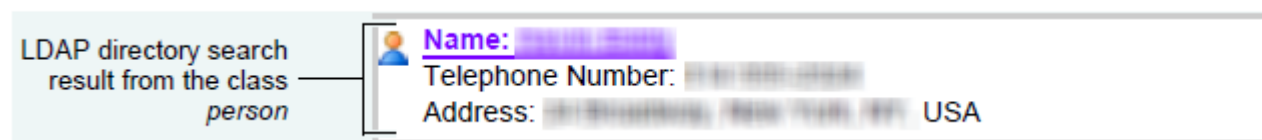
### LDAP Classes and Attributes

Exalead CloudView allows you to select the attributes of every class to display in the search results. For example, when you configure an LDAP class **person**, you also select the attributes to be crawled: **name**, **telephone**, **address**, and **memberOf**.

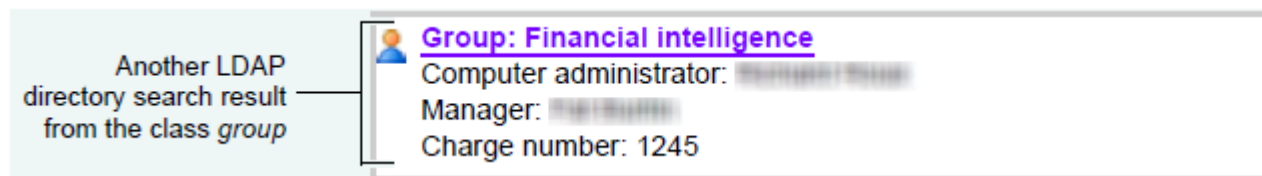
Likewise, when you configure an LDAP class **group**, you can also select the attributes to be crawled such as: **name**, **administrator**, **manager**, and **chargeNumber**.

person	group
-name	-name
-telephone	-administrator
-address	-manager
	-chargeNumber

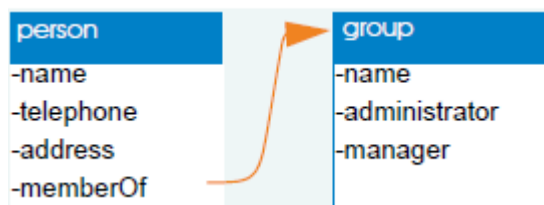
A search for a person in this LDAP directory provides the following hit content:



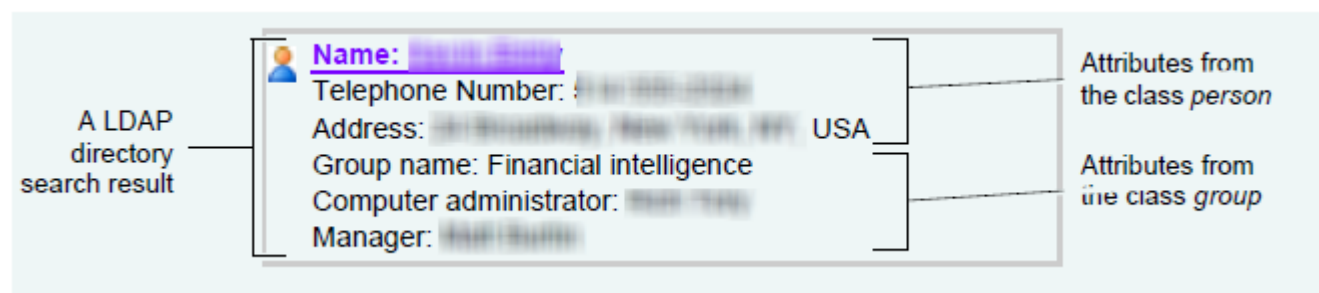
A search for a group in this LDAP directory provides the following hit content:



Now let us see a second scenario from the same LDAP directory that includes a reference. As shown below, an attribute may reference another class: **memberOf** that references **group**. In this case, you also select the attributes that are to be crawled from **group**: **name**, **administror**, and **manager**. When searching for a person, we are not interested in the charge number for their group, so this attribute is not to be indexed.

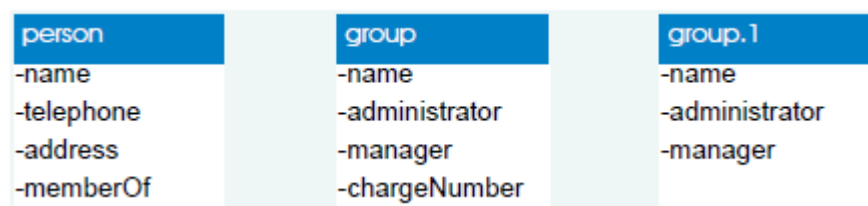


A search for a person in this LDAP directory provides the following search result content:



## Add LDAP References

To add LDAP references in Exalead CloudView, you must configure LDAP classes in the administration interface.



In this example, **group.1** is a copy of **group**, but without the attribute **chargeNumber**. This is the class that **memberOf** will reference.

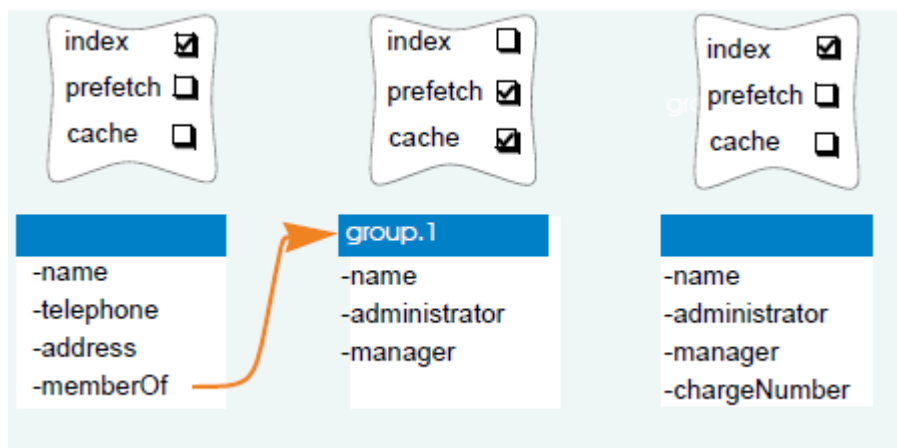
Then make a reference in class **person**, attribute **memberOf** to class **group.1**.

## Classes to Index

To follow the above example, only the root classes must have the **index** parameter defined.

To follow the above example, only the root classes must have the **index** parameter defined.

Classes that are only referenced must not have the **index** parameter defined.



Referencing a class can induce a lot of requests to the LDAP server. To avoid this, you can select **Prefetch** and **Cache**.

- **Prefetch** allows the connector to get all the class entries before indexing time with one big request, and cache it once and for all. No more requests will be required for this class.
- **Cache** allows the connector to cache the class entries during indexing time to avoid multiple requests.

## Configure the LDAP Connector

### Configure the Connection to the LDAP Server

Configure the connection to the LDAP server by defining the following properties from the **Configuration** tab.

1. In **Host**, specify the name of the host that runs the LDAP server.
2. In **Port**, specify the port on which the LDAP service is accessible.
3. In **Username**, specify the user name used to access the LDAP service.
4. In **Password**, specify the password used to access the LDAP service.
5. In **Base**, specify the distinguished name that identifies the entry point where the search is to be started.



For example `cn=Users, dc=office, dc=exalead, dc=com`

6. In **Creation timestamp name**, specify the name of the attribute in which to find the creation timestamp.

A timestamp shows when a record was created or modified. Exalead CloudView uses the timestamp to crawl only attributes that have been created or modified since the last crawl.

7. In **Modification timestamp name**, specify the name of the attribute in which to find the modification timestamp.
8. To filter data from the LDAP directory, add an LDAP query filter script in **LDAP filter**.

For example, `(|(ou=Human Resources)(ou=Financial Services))` excludes all LDAP entries that are not from Human Resources or Financial Services as specified in the Organizational Unit (ou) fields.

9. Click **Apply**.

## Specify the LDAP Classes and Attributes

This is where you configure the LDAP attributes to crawl. You select the LDAP attributes to retrieve from the LDAP directory, and specify an additional filter that refines the LDAP query.

To configure this, you need a thorough understanding of LDAP and LDAP schemas, see [LDAP Classes and Attributes](#).

1. In **Class config**, click **Add item**.
2. In **Item 0**, set up the LDAP class that Exalead CloudView must access.
  - a. In **Name**, enter an internal name for this class. This is used as the name of the category to be displayed in the search results for this class, for example, `People`
  - b. Select **Index** to index the class. See [Classes to Index](#).
  - c. In **LDAP class name**, enter the class name. For example, `person`. See [LDAP Classes and Attributes](#).
3. In **Attributes**, click **Add item**.
4. Set up the LDAP attributes to crawl for the above class.
  - a. In **LDAP attribute name**, enter the LDAP attribute name. For example, `name` for the `person` class.
  - b. In **Meta name**, enter the meta name to be used for this attribute. For example, `Name`.
5. Add more classes and attributes.
6. Click **Apply** to save and apply the new connector configuration.

## Parameter Descriptions

### Global Configuration Parameters

Field	Description	Default
<b>Host</b>	The name of the LDAP host.	
<b>Port</b>	The port to use for the LDAP connection.	
<b>Username</b>	Name of a user that has full read access on the LDAP server.	
<b>Password</b>	Password of the defined user.	
<b>Authentication method</b>	The LDAP implementation method to use: <code>simple</code> , <code>kerberos41</code> , <code>kerberos42</code> , <code>digest</code> , <code>ntlm</code> , or <code>negotiate</code> .	<code>simple</code>
<b>LDAP protocol</b>	The LDAP protocol to use; <code>ldapv2</code> or <code>ldapv3</code> .	<code>ldapv3</code>
<b>Base</b>	Specifies the distinguished name (DN) that identifies the entry point where the search starts.  You can override it at the class level.	
<b>Scope of the search</b>	Search scope: <code>base</code> , <code>onelevel</code> , or <code>subtree</code> .	<code>subtree</code>
<b>Encryption mode</b>	The encryption mode to use: <code>none</code> , <code>ssl</code> , or <code>starttls</code> .	<code>none</code>
<b>Timeout (ms)</b>	Timeout for the connection in milliseconds.  If this value is <code>-1</code> , there is no timeout.	
<b>Ignore case</b>	Indicates whether LDAP attributes names matching must be case insensitive.	<code>false</code>
<b>Creation timestamp name</b>	Creation timestamp attribute name. When specified, it allows incremental indexing.  You can override it at the class level.	
<b>Modification timestamp name</b>	Modification timestamp attribute name. When specified, it allows incremental indexing.  You can override it at the class level.	
<b>LDAP filter</b>	An optional LDAP query filter script.	

Field	Description	Default
	You can define it at the class level.	
<b>Ignore SSL cert</b>	Ignores the SSL certificate used to secure the LDAP server.	true

## Class Config Parameters

Field	Description
<b>Name</b>	The internal name for the class. This name displays as a facet in search results.
<b>LDAP class name</b>	The LDAP <b>class</b> name. For example, the <code>person</code> class (see <a href="#">LDAP Classes and Attributes</a> ).
<b>Index</b>	The Boolean value that determines whether to index the class. The default value is true. See <a href="#">Classes to Index</a> .
<b>Prefetch</b>	The Boolean value that determines whether to prefetch all the entries of this class before the indexation. Prefetch is useful to improve performance for classes that are referred by another class.
<b>Cache</b>	The Boolean value that determines whether to cache the entries of this class if we encounter them, to avoid fetching them multiple times. Cache is useful to improve performance for classes that are referred by another class.

## Attribute Parameters

Field	Description
<b>LDAP attribute name</b>	The LDAP attribute name, for example, <code>memberOf</code> . If null, the DN is used.
<b>Meta name</b>	The meta name to use for this attribute, for example, <code>Group</code> .
<b>Meta name suffix</b>	Suffixes the meta name with # followed by the DN of the current entry. This option has no effect when the <b>Tree concatenation</b> option is selected.
<b>Join class</b>	The class that this attribute refers to (one of the defined class references, for example, <code>group</code> ). See <a href="#">Add LDAP References</a> .
<b>Join attribute</b>	Optional. The LDAP attribute of the referred "Class" to perform the join.

Field	Description
	If null, the DN is used.
<b>Tree concatenation</b>	<p>Concatenates the meta value to push on the tree branches of the LDAP entries used in the document.</p> <p>To configure the concatenation parameters (order, separator, etc.), add a Tree Concatenation Config with the same meta name than this attribute.</p>

# Logs Connector

This section describes the functionality and configuration of the Logs connector.

## Logging Framework

### Configure the Logs Connector

### Use Case

## Logging Framework

For each file structure, you may choose to group specific fields using the **Group on** column.

The log connector definition is based on the log file structure below:

### log4j

You can use this format to index logs in log4j format.

**Example:** [2019/11/19-14:18:49.590] [info] [Thread-13]  
[replication.manager] execute command on 0/i0: getFilesInUse (serial=5)

5 log fields are required:

- **date:** [2019/11/19-14:18:49.590]
- **level:** [info]
- **thread:** [Thread-13]
- **logger:** [replication.manager]
- **message:** execute command on 0/i0: getFilesInUse (serial=5)

### Apache

You can use this format to index logs in apache format.

**Example:** 127.0.0.1 - frank [10/Oct/2019:13:55:36 -0700] "GET /  
apache\_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html"  
"Mozilla/4.08 [en] (Win98; I ;Nav) "

13 log fields are required:

- **ip:** 127.0.0.1
- **identid:** -

- **userid:** frank
- **date:** 10/Oct/2019
- **time:** 13:55:36
- **timezone:** 0700
- **method:** GET
- **requested:** /apache\_pb.gif
- **protocol:** HTTP/1.0
- **statuscode:** 200
- **size:** 2326
- **referer:** http://www.example.com/start.html
- **useragent:** Mozilla/4.08 [en] (Win98; I ;Nav)

See <http://httpd.apache.org/docs/trunk/en/logs.html> for more details on Apache log format.

## Auto

You can use this format to manage log structure automatically.

Configure the delimiters used to define the log structure in the **Delimiters** section (for example, [ and ]).

## Custom

You can use this format to manage log structure using a regular expression in the **Regular expression** field.

Each capturing group must correspond to a log field defined in the **Log fields** section.

## Configure the Logs Connector

This section implies that the connector has already been added.

See [Creating a Standard Connector](#).

## Configure the Connector

1. On the Administration Console home page, under **Connectors**, select the log connector you require.

The configuration screen displays for the connector.

2. Configure the connector's main parameters.

Parameter	Description
<b>Scan latest entries only</b>	If selected, the next scan starts at the latest scanned log entry.
<b>Line format</b>	Specify the format of the log files. According to the selected pattern, you need to create the corresponding log fields.
<b>Encoding</b>	Enter the file encoding to use. By default, <b>utf-8</b> .
<b>Max number of crawled lines</b>	Defines the maximum number of lines crawled for a single file.
<b>Log fields</b>	Add fields corresponding to the selected log file format in <b>Line pattern definition</b> .
<b>Additional output</b>	<p>You can:</p> <ul style="list-style-type: none"> <li>• Add a meta for the name of the file that has been scanned: <b>Set file name in the meta</b>.</li> <li>• Add a meta for the line scanned in the file: <b>Set file line in the meta</b>.</li> <li>• Add a meta for data that have not been parsed properly: <b>Set lines that do not match the format in the meta</b>.</li> <li>• Group data that has not been parsed properly in a single document: <b>Group consecutive lines that do not match the format</b>.</li> </ul>
<b>Filesystem paths</b>	<p>Enter the filesystem path for the files (.ZIP, .LOG, .GZ) or directory to crawl. You can prefix paths by:</p> <ul style="list-style-type: none"> <li>• <code>data://</code> OR <code>/data</code> – relative to the <code>&lt;DATADIR&gt;</code></li> <li>• <code>resource://</code> – relative to the <code>NGRESOURCEPATH</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li>• <code>kit://</code> – relative to the <code>&lt;INSTALLDIR&gt;</code></li> <li>• <code>file://</code> – absolute path, for example, <code>file://temp/csv-data/</code></li> </ul> <p>Click <b>Add path</b> for each path you want to crawl.</p>

3. You are ready to check your connector's configuration.

See how [Check the Logs Connector Config](#).

## Check the Logs Connector Config

1. In the **Test** section, enter a log file entry corresponding to the log file format selected in **Line format**.
2. Click **Check syntax** and **Apply**.

If the check is successful, a green "**Matching is OK**" message is displayed. You are now ready to scan and index your documents.

## Use Case

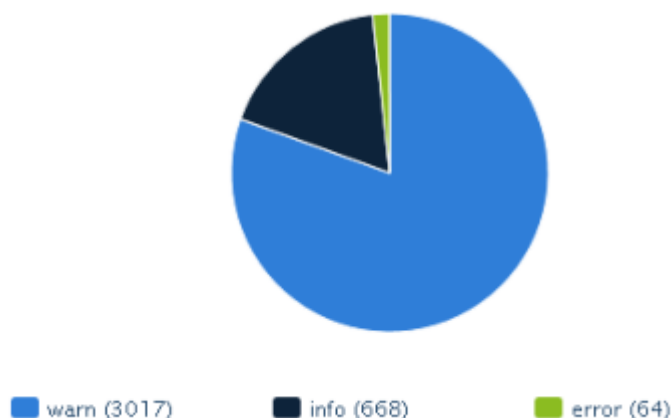
In the following example, we first analyze the number of log levels displayed in a .LOG file using a pie chart. We then see how to refine the connector configuration to display more relevant data.

This section implies that you are familiar with the Mashup Builder and Administration Console components.

### Step 1: Analyze Log Levels

1. We want to analyze a .LOG file: configure your connector with the **log4j** line format.
2. We want to display a pie chart sorted on log levels:
  - a. In the Administration Console, set the log level as category facet in your data model.
  - b. In the Mashup Builder, configure a pie chart based on the facet defined previously in your **search** page.
3. Reindex.

You get the following pie chart on your search page:



Many errors appear. Let us see how we can refine our analysis.



## Step 2: Group Stack Traces to Refine Analysis

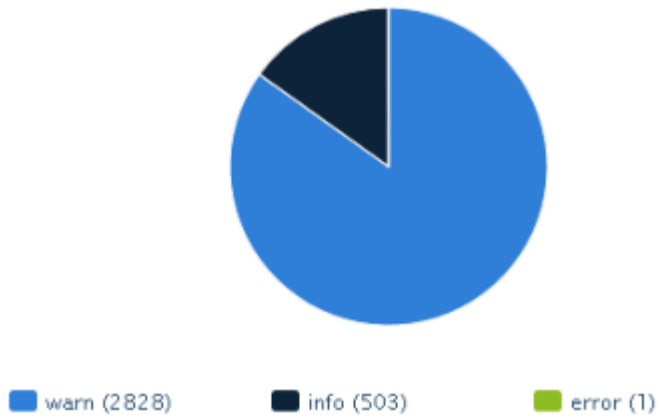
Our log file displays the following error:

```
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]/
mashup-builder/mashup-ui-proxy/default/staging-builder/BzRVxU6K/downloads:
java.lang.IllegalStateException: STREAM
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]at
org.eclipse.jetty.server.Response.getWriter(Response.java:683)
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]at
org.apache.jasper.runtime.JspWriterImpl.initOut(JspWriterImpl.java:191)
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]at
org.apache.jasper.runtime.JspWriterImpl.flushBuffer(JspWriterImpl.java:184)
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]at
org.apache.jasper.runtime.JspWriterImpl.write(JspWriterImpl.java:458)
[2014/03/14-14:17:29.129] [error] [1731984037@jtp-adminui-443]
[org.eclipse.jetty.servlet.ServletHandler]at
org.apache.jasper.runtime.JspWriterImpl.write(JspWriterImpl.java:471)
.....
```

With our current connector configuration, each row containing `[error]` is analyzed as an error. In this example, we have in fact one single error followed by the details of the exception.

1. We must configure our connector to ignore the stack trace: in your connector configuration, select **Group on** for the following fields:
  - date
  - level
  - thread
  - logger
2. Reindex.

You get the following pie chart on your search page:



Rows related to the same error have been grouped.

# Managed Push API Connector

This section describes the functionality and configuration of the Managed Push API connector to use Push API Filters with unmanaged connectors.

[Introducing the Managed Push API Connector](#)

[Configuring the Managed Push API Connector](#)

## Introducing the Managed Push API Connector

The Managed Push API connector allows you to use Push API filters on unmanaged connectors. You can use it for unmanaged connectors that are not written in Java.

**Recommendation:** If your unmanaged connectors are written in Java, use the RScan client connector.

The Managed Push API connector listens to the PAPI servlet on a dedicated PAPI server, and:

- Waits for the documents pushed by the unmanaged connectors, since it starts listening continuously when Exalead CloudView starts.
- Lets you decide whether documents are owned by the unmanaged connector pushing to the Managed Push API Connector, or by the managed connector itself, if you select the **Keep source ownership** property.
- Supports the **Abort scan** operation, which stops the PAPI server (documents can no longer be sent).

When the connector owns the documents it pushes, the "fetch" operation can be supported with an external fetch service in the internal configuration:

```
<connect:SourceFetchConfig
customFetcherUrl="FETCH_SERVICE_URL"
fetchProtocol="PROTOCOL_VERSION"
allowRawDocumentFetch="true"
/>
```

**Note:** For more information, see "Push API filters" in the Exalead CloudView Connector Programmer's Guide.

## Configuring the Managed Push API Connector

Follow the procedure below to configure the managed Push API connector.

This section implies that the connector has already been added. See [Creating a Standard Connector](#).

1. Configure the following options.

Parameter	Description
<b>PAPI port</b>	Specifies the network port of the Push API server run by this connector. Specify the same port for the unmanaged connectors that must push documents to this connector.
<b>Keep source ownership</b>	<p>If selected, Exalead CloudView considers that documents pushed by this connector are owned by the real source connector. Otherwise, it considers this connector as the owner of all pushed documents, and the fetch operation is not supported.</p> <p><b>Recommendation:</b> Select this option if the source connectors implement the <code>fetch</code> operation. In that case, the source connectors and this managed connector must have different names so that Exalead CloudView can distinguish them.</p>
<b>No. threads</b>	<p>Specifies the number of threads available for the Push API server.</p> <p>If you specify a value inferior or equal to 0, the Push API server creates as many threads as required, but reuses available ones.</p>

2. Click **Apply**.

You do not have to click **Scan** for this connector as it scans document continuously.

3. Go to the **Deployment** tab and disable the **Buffer operations** property.

When you add a managed connector, a buffer is implicitly added to batch operations between the connector and the indexing server. The buffer is flushed from time to time at the end of the connector scan.

This is however problematic with the Managed Push API Connector as its scan never ends, and Push API operations can remain stuck in the buffer. That is why we recommend disabling the **Buffer operations** property.

# RScan Client Connector

This section describes the functionality and configuration of the RScan Client connector.

[Introducing the RScan Client Connector](#)

[Configuring the RScan Client Connector](#)

## Introducing the RScan Client Connector

The RScan Client connector allows you to start operations and use Push API filters on unmanaged Java connectors. To do so, it talks with the Remote Scan server, which must reference these unmanaged connectors.

### Before You Start

To use the RScan server, you must copy the following jar file packaged within the Exalead CloudView kit on the remote server: `datainteg-java-commons.jar`

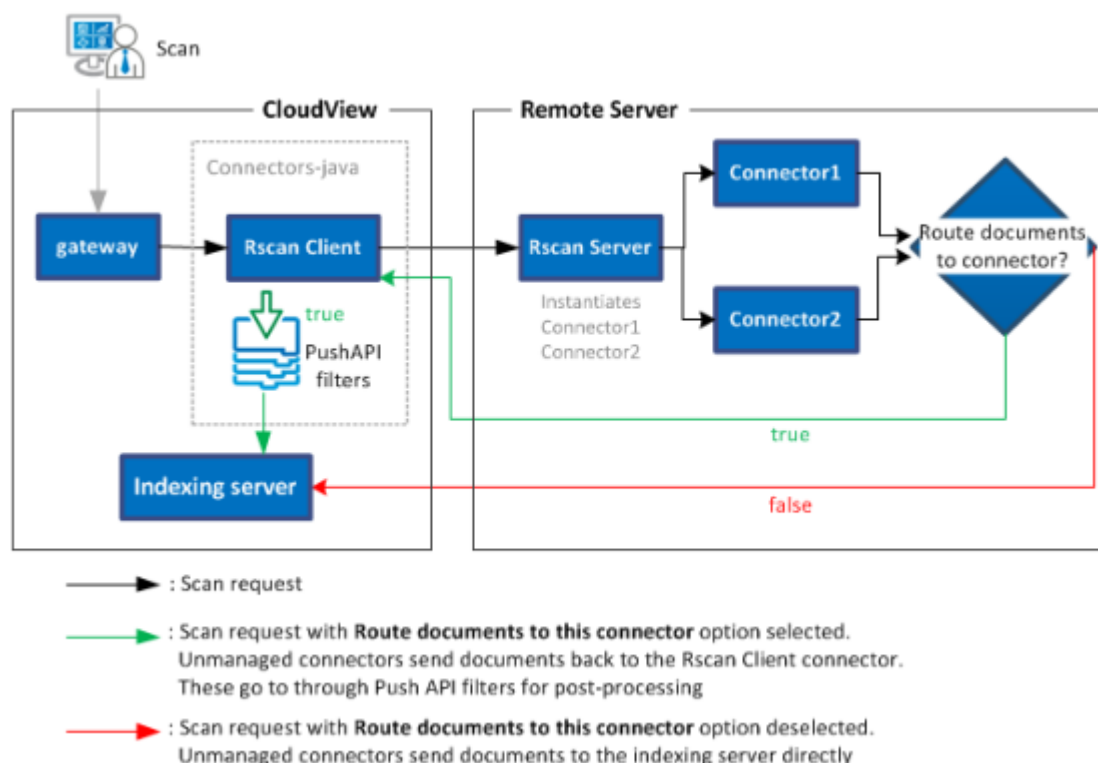
Developing unmanaged connectors requires the following jar files:

- `datainteg-java-commons.jar` as the RScan server is a library used for developing unmanaged connectors.
- `cloudview-java-plugin.jar`
- `papi-java-client.jar`
- `papi-java-connector.jar`

Once the RScan server is installed, edit it to instantiate your unmanaged connectors.

**Note:** For more information, see "Deploying Connectors on a Remote Server" in the Exalead CloudView Connector Programmer's Guide.

## RScan Global Architecture



## Configuring the RScan Client Connector

This section implies that the connector has already been added.

See [Creating a Standard Connector](#).

1. Configure the following options.

Parameter	Description
<b>Host</b>	Specifies the server hosting the remote scan service.
<b>Port</b>	Specifies the network port where the remote RSCAN service is listening.
<b>URL prefix</b>	<p>Specifies the URL prefix used to reach the RSCAN remote servlet. For example, with the following settings:</p> <ul style="list-style-type: none"> <li>• <b>Host:</b> <code>myserver.com</code></li> <li>• <b>Port:</b> <code>5000</code></li> <li>• <b>URL prefix:</b> <code>path/to/service</code></li> </ul> <p>The RSCAN servlet URL is:</p> <p><code>http://myserver.com:5000/path/to/service/rscan/</code></p>

Parameter	Description
<b>Route documents to this connector</b>	<p>Routes the documents of your remote unmanaged connectors back to the RScan Client connector instead of sending them to the indexing-server directly. This option is required if you want the documents pushed by unmanaged connectors, to go through the RScan Client connector PushAPI filters.</p> <p>Disable this option if you do not use Push API filters and want to keep high performances (CPU...).</p>
<b>Use configuration of existing connector</b>	<p>Sends the specified connector configuration to the remote connector. If the configuration changes in the Administration Console, the new configuration is sent to the remote connector at the next full scan operation.</p>
<b>Retrieve fetch protocol</b>	<p>Specifies the fetch server's protocol:</p> <ul style="list-style-type: none"> <li>• <b>v1</b> – Exalead CloudView Retrieve Protocol V1</li> <li>• <b>v2</b> – Exalead CloudView Retrieve Protocol V2</li> <li>• <b>rpv3</b> – Exalead CloudView Retrieve Protocol V3</li> <li>• <b>networkRetriever</b> – Use Exalead Web Crawler. If specified, the <b>Document server URL</b> option must be stored as publicName fetcherName</li> </ul>

2. Optionally, to retrieve the configuration parameters of your remote unmanaged connector, click **Retrieve default**.
3. Click **Apply**.

You can now start scan operations and use Push API filters on your unmanaged connectors.

# Replay Connector

This section describes how to use and configure the Replay server and connector in Exalead CloudView.

[Introducing the Replay Connector](#)

[Configuring the Replay Connector](#)

## Introducing the Replay Connector

The Replay connector allows you to repush data from a given source. You can also specify a subset of the scan or several scans at once using a given time frame.

The Replay components (Replay Server and Replay Connector) are designed for two main purposes:

- Capture a scan data flow from a source to be able to replay it without crawling the source once again.

This is useful in test and development environment when you need to clear index and sources several times or when running unit tests on a consistent corpus.

- Use the replay storage to persist documents that may need to be repushed inside an index and that can be deleted from the original source (for example, when dealing with historical data).

## Configuring the Replay Connector

The Replay service has 3 components embedded by default in Exalead CloudView:

- The Replay server role
- The Replay connector
- The Replay PushAPI Filter

### Deploy the Replay Server Role

1. Go to **Deployment > Roles**.
2. Click **Add roles**.
3. Select **Data integration > Replay server**.
4. In the newly created Replay server, enter:
  - a. The **Instance name**, for example, `replay0`.



- b. The **Connection idle time** (in seconds)
- c. The **Port** used by the Replay HTTP server, for example, 8080.
- d. In addition, to use an external database to retrieve and write replay data, you must configure the following options:

Option	Description
<b>Engine type</b>	Choose between Oracle, SQLite, and PostgreSQL.  For Oracle and PostgreSQL, the JDBC driver jar file must be available in the <DATADIR>/javabin directory.
<b>Connection URL</b>	The URL of the database used to retrieve and write replay data. Use the following URL formats: <ul style="list-style-type: none"> <li>For Oracle: jdbc:oracle:thin:@&lt;hostname&gt;:&lt;port Number&gt;:&lt;databaseName&gt;</li> <li>For PostgreSQL: jdbc:postgresql://host:port/database</li> <li>For SQLite: this parameter is ignored.</li> </ul>
<b>Database Login</b>	The login of a user allowed to access the database, read the data model and create tables.  For Oracle and PostgreSQL
<b>Database Password</b>	The password associated to the login.  For Oracle and PostgreSQL
<b>Overlay threshold (in Mb)</b>	Size threshold over which a PAPI command is not stored in the database, but copied in a separate file and referenced by the database.  If you specify 0 or a negative value, the command is always stored in the database.  <b>Recommendation:</b> Specify 0 for Oracle and PostgreSQL.
<b>Transaction size threshold (in Mb)</b>	Size threshold of the database transactions gathering PAPI commands. If a transaction exceeds this value, it is committed to the database.  If you specify 0 or a negative value, the transaction is committed to the database for each PAPI command.  <b>Note:</b> If a problem occurs (power outage, network failure, etc.), each command that is part of a non-committed transaction is lost.

Option	Description
<b>Transaction duration threshold (in seconds)</b>	<p>Maximum duration during which a database transaction cannot be committed.</p> <p>If you specify 0 or a negative value, the transaction is committed to the database for each PAPI command.</p> <p><b>Note:</b> If a problem occurs (power outage, network failure, etc.), each command that is part of a non-committed transaction is lost.</p>

- Click **Apply**.
- Restart Exalead CloudView.

## Capture a Data Flow

### Configure the Connector to Replay

You can add a PushAPI filter to your connector configuration and define the Replay server to use.

- Edit the connector you want to replay.
- Display the **Advanced** tab.
- In **PushAPI filters**:
  - For **PushAPI filter type**, select **Replay PushAPI filter**.
  - In **Replay server instance**, enter the **instance name** you defined previously for the Replay server role (for example, `replay0`).
  - In **Name of this source**, enter the name of the connector you want to replay (for example, `mysource`).

**Note:** This name is also asked for the **Source to replay** option in the next section, dedicated to the connector configuration.

- Scan your connector.

**Note:** The number of documents displayed for a connector applies to all connectors sharing the same Replay server.

### Create the Replay Connector

- Go to **Index > Connectors**.
- Click **Add connector**.
- Select the **Replay** connector type and give it a name.
- Click **Accept**.

5. Configure the connector:
  - a. In **Source to replay**, enter the name of the connector you want to replay (for example, `mysource`).
  - b. In **Replay Server instance**, enter the name of the Replay server defined previously (for example, `replay0`).
  - c. Select a time frame for the indexing you want to replay. If no time frame is defined, it pushes everything.
6. Select the PAPI commands you want to replay in **Select the PAPI commands to replay**. By default, all PAPI commands are sent.

# XML Connectors

This section describes the functionality and configuration of the XML Connectors.

[Introducing the XML Connector](#)

[Configuring the XML Simple Connector](#)

[Configuring the XML Advanced Connector](#)

[Developing a Custom XML Processor](#)

[Installing Custom Code](#)

## Introducing the XML Connector

The XML connector allows you to index XML files stored on a filesystem. It can extract data from XML files in several ways using predefined processors such as XPath, XSLT, by element name.

You can also develop your own processor and use it with this connector.

The XML connector comes in the following variants:

- The **XML (simple)** connector for rapid configuration.
- The more powerful **XML (advanced)** connector, which allows you to run several processors on XML documents. Some processors can be chained, to feed one processor with the result of another processor.

**Important:** Namespaces are not taken into account when XML files are parsed.

## Configuring the XML Simple Connector

The XML connector allows you to extract data from XML documents in several ways. This section implies that the connector has already been added. See

[Creating a Standard Connector](#).

[Extract by XPath Method](#)

[Extracting by XML Elements Method](#)

[Extracting Using XSLT Method](#)

[Using the Split XML Documents Method](#)

[Property Descriptions](#)

## Extract by XPath Method

This extraction method associates the results of the XPath selection to metas.

Configure the parameters:

- **XPaths:** list of XPath mappings
  - **Meta:** the name of meta associated with the XPath results
  - **XPath:** the XPath expression

## Extract by XPath

1. In **Configuration**, keep the global configuration default settings.
2. Expand **Root Paths**, and click **Add item** to add a path.
3. In **Root path**, enter the file or folder path to index.  
For example, `/data/<USER>/xml-sample-data/contacts`
4. Expand **XPaths**, and click **Add item**.
  - a. In Item n, enter the Xpath expressions to use. For example, `/person/@name` to retrieve the content of the `name` attribute, and `text()` to retrieve the content of a node, like a phone number.
  - b. Enter the meta names associated with the content in **Meta**. For example, `title` and `text`.
5. Click **Apply** to apply changes to the configuration.

The connector configuration is complete. You can now scan and index the documents.

## Example

Let us consider the following document:

```
<root>
<D>
<M N="uri">050b-180b-536cc63c-000000040387-00</M>
<M N="session_dest_city">mountain view</M>
<M N="session_dest_country">us</M>
<M N="title">Nursing Portfolio</M>
</D>
<D>
<M N="uri">0203-6dd1-5369d61f-00000004e0b83-00</M>
<M N="session_dest_city">mountain view</M>
<M N="session_dest_country">us</M>
<M N="summary">##Subject: Proven in clinical trials</M>
</D>
</root>
```

If we configure in:

- **XPATH split processors**, an item set to `/root`
- **XPath processors**, the following Xpath items:

Xpaths item	configuration
uri	meta: uri XPath: <code>/D/M[@N='uri']</code>
first attribute	meta: session_dest_city XPath: <code>/D/M[@N='session_dest_city']</code>
second attribute	meta: session_dest_country XPath: <code>/D/M[@N='session_dest_country']</code>
third attribute	meta: summary XPath: <code>/D/M[@N='summary']</code>

The following metas are added to the document:

```
uri:0203-6dd1-5369d61f-0000004e0b83-00
session_dest_country:us
session_dest_city:mountain view
summary:##Subject: Proven in clinical trials
```

## Extracting by XML Elements Method

You can use the XML Elements to perform a by-name node (or attribute) selection and push their value as metas. The connector selects nodes or attributes based on the include and exclude lists.

The following rules apply:

- Children of included nodes are pushed as separate metas.
- Children of excluded nodes are completely ignored (as well as their attributes).

You can configure the parameters:

- **Include elements**: list of element mappings.
  - **Meta**: name of meta associated with the node content
  - **Element**: name of the XML element to extract. This can also represent an attribute when specifying a string in the form `node@attribute`. For example: `book@title`.
- **Exclude elements**: list of node names to skip

When no Meta is specified in **Include elements**, the connector uses the name of the element or attribute to infer the meta name. For example:

- If you set **Element** to `book`, the node text is pushed in the `book` meta.
- If you set **Element** to `book@title`, the attribute text is pushed in the `title` meta.

This procedure describes how to extract by Element name

1. In **Configuration**, keep the global configuration default settings.
2. Expand **Root Paths**, and click **Add item** to add a path.
3. In **Root path**, enter the file or folder path to index.

For example, `/data/<USER>/xml-sample-data/contacts`

4. Expand **Include elements**, click **Add item** and enter the element mapping to include.
  - a. Enter the name of the XML node to extract in **Element**. For example, `person@name`.
  - b. Enter the meta name associated with the node content in **Meta**. For example, `title`.
5. Click **Add item** and repeat Step 4 for each mapping required.
6. Enter the list of element mappings to exclude in **Exclude elements**.

For this example, you can leave the settings as is.

7. Click **Apply** to apply changes to the configuration.

The connector configuration is complete. You can now scan and index the documents.

## Extracting Using XSLT Method

This processor performs XSL transformations on XML documents and passes the result to the extraction methods.

In the top pane, configure the **Stylesheet** parameter to define the stylesheet file path. The connector transforms XML files according to this stylesheet.

You can then use one of the following extraction methods:

- If the transformation result returns an XML file with the following `PAPI_document` format, then appropriate metas are automatically pushed:

```
<PAPI_document>
  <PAPI_meta name="name">value</PAPI_meta>
  <!-- ... -->
</PAPI_document>
```

For example,

```
<PAPI_document>
  <PAPI_meta name="text">Sally M.</PAPI_meta>
  <PAPI_meta name="phone">555 1234</PAPI_meta>
</PAPI_document>
```

- Xpath expressions - if **XPath after XSLT** is selected, extraction is performed on the XSLT result instead of the original document or chunk.
- by Element name - If **Elements extraction after XSLT** is selected, by-name extraction is performed on the XSLT result instead of the original document or chunk.

In this example, the connector transforms the XML file and then extracts using the PAPI document method.

1. In the global configuration pane, for **Stylesheet**, enter the stylesheet to use for transformation.

For example, `/data/<USER>/xml-sample-data/contacts.xml`

2. Expand **Root Paths**, and click **Add item** to add a path.

3. In **Root path**, enter the file or folder path to index.

For example, `/data/<USER>/xml-sample-data/contacts`

4. Click **Add item** to add more paths to the list of **Root Paths**.

5. Click **Apply**.

## Using the Split XML Documents Method

You can enable the **Split** parameter to split XML documents in several chunks.

Every chunk is then handled as a separate document. Every first-level child of the XML document is extracted.

For example, if you want to index all the contacts from a single `contacts.xml` file:

```
<contacts>
  <person name="John M.">
    <phone>2222 3333</phone>
    <folder>work</folder>
  </person>
  <person name="Mary K.">
    <phone>1000 1000</phone>
    <folder>work</folder>
  </person>
  <person name="Sally M.">
    <phone>4242 4242</phone>
    <folder>work</folder>
  </person>
  <!-- ... -->
</contacts>
```

Each `<person>` node represents a document and stands as the root node when configuring expressions. For example, to retrieve the content of the name attribute with a XPath expression, you can enter: `/person/@name`



1. In the global configuration pane, set **Split** to **true**.

The connector splits the XML document into several chunks.

2. Expand **Root Paths**, and click **Add item** to add a path.

3. In **Root path**, enter the file or folder path to index.

For example, `/data/xml-sample-data/contacts/contacts.xml`

4. Configure the extraction parameters. See the details for the supported methods:

- [Extract by XPath](#)
- [Extracting Using XSLT Method](#)

5. Click **Apply**.

## Property Descriptions

### Global Configuration Properties

The following global configuration properties are available for the XML simple and the XML advanced connectors.

The properties flagged with [R] are required.

Type	Property	Value
Filesystem	<b>File extensions</b>	Specifies the file extensions to include in the crawl and index processes.
	<b>Recursive</b>	Indexes the subdirectories of the path. Default is <code>true</code> .
	<b>Index names</b>	[R] Indexes non-matching extensions. Default is <code>false</code> .
	<b>Max. input size</b>	[R] Specifies the maximum size of files. Default is <code>10MB</code> .
	<b>Push folders as documents</b>	Pushes folders as documents.
	<b>Max. document queued</b>	Specifies the maximum number of documents in queue. Default is <code>1000</code> .
	<b>Max. folder queued</b>	Specifies the maximum number of folders in queue. Default is <code>1000</code> .

Type	Property	Value
	<b>No. pipeline document thread</b>	Specifies the number of threads handling documents. Default is 4.
	<b>No. pipeline folder thread</b>	Specifies the number of threads handling folders. Default is 4.
XML	<b>Normalize meta names</b>	Normalizes meta names (meta names are put in lowercase and spaces are replaced by an underscore). Default is <code>false</code> .
	<b>Incremental</b>	Pushes only new files from the filesystem. Default is <code>false</code> .
	<b>Push XML as meta</b> (for XML simple only)	Pushes the XML document (or the chunk if the document is split) as a meta named <code>xmlbody</code> . Default is <code>false</code> .
	<b>Verbose</b>	Enables the verbose mode. Default is <code>false</code> .
	<b>Entry point id</b> (for XML advanced only)	[R] Specifies the first processor of the chain. The connector needs to know where the chain of processors begins so an <b>Entry point id</b> has to be supplied.

## Extraction Properties

The following extraction properties are available for the Simple XML connector. Some properties are located in the global configuration section.

The properties flagged with [R] are required.

Property	Value
<b>Root Paths</b>	<p>Specifies the list of files or folders to index.</p> <p>You can prefix paths by:</p> <ul style="list-style-type: none"> <li><code>data://</code> OR <code>/data</code> – relative to the <code>&lt;DATADIR&gt;</code></li> <li><code>resource://</code> – relative to the <code>NGRESOURCEPATH</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li><code>kit://</code> – relative to the <code>&lt;INSTALLDIR&gt;</code></li> </ul>

Property	Value
	<ul style="list-style-type: none"> <li>• <code>file://</code> – absolute path, for example, <code>file://temp/csv-data/</code></li> <li>• <code>run://</code> – relative to the <code>NGRUNDIR</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> <li>• <code>config://</code> – relative to the <code>NGCONFIGDIR</code> (see definition on <code>&lt;DATADIR&gt;/bin/ngstart.env</code>)</li> </ul>
<b>Root path</b>	Specifies either a file or a folder.
<b>Split</b>	<p>Determines whether to consider every first-level child as a separate document.</p> <p>Default is <code>false</code>.</p>
<b>Stylesheet</b>	Specifies the XSL Transformation file to use.
<b>XPath after XSLT</b>	<p>Enables XPath extraction on XSL transformation result</p> <p>Default is <code>false</code>.</p>
<b>Element extraction after XSLT</b>	<p>Enables element name extraction on XSL transformation result.</p> <p>Default is <code>false</code>.</p>
<b>Include elements</b>	<p>For extraction by Element name</p> <p>Specifies a list of elements to include:</p> <ul style="list-style-type: none"> <li>• <b>Element</b> [R]: name of the XML node to include</li> <li>• <b>Meta</b>: name of the associated meta</li> </ul>
<b>XPaths</b>	<p>For extraction by XPath.</p> <p>Specifies a list of XPath expressions. Each result of XPath extraction is associated with a meta:</p> <ul style="list-style-type: none"> <li>• <b>XPath</b> [R]: an XPath expression</li> <li>• <b>Meta</b> [R]: name of the associated meta</li> </ul>
<b>Exclude elements</b>	Specifies a list of elements to exclude.

## Configuring the XML Advanced Connector

This connector allows you to run several processors on XML documents. Some processors in the pipeline can be chained, that is to say that the result of one processor is fed to another. Processors can be split into two categories:

- Regular processors that handle one document and extract data from it.
- Split processors that extract chunks of XML data from one (possibly large) document and delegate chunk processing to other processors.

This section details the processors bundled with the connector. These are all regular processors unless stated otherwise.

## About the Processor Pipeline

### Configuring the PAPI Document Processor

### Configuring the XML Element Processor

### Configuring the XPath Processor

### Configuring the XSLT Processor

### Configuring the Tee Processor

### Configuring the XML Attach Processor

### Configuring the Child Split Processor

### Configuring the XPath Split Processor

### Configuring the Custom Processor

## About the Processor Pipeline

To configure the XML advanced connector, you need to:

- Configure the Global Configuration properties, see [Property Descriptions](#) .
- Create a pipeline with the processors described in this section.

In the pipeline:

- Every processor has an id.
- The connector needs to know where the chain of processors begins, so an **Entry point id** has to be supplied.
- If a processor is not the entry point, and not referenced by another processor, it will never be run.

## Configuring the PAPI Document Processor

This processor extracts data from documents that look like:

```
<PAPI_document>
<PAPI_meta name="name">value</PAPI_meta>
<!-- ... -->
</PAPI_document>
```

1. Expand **PAPI document processors**, and click **Add item**.
2. In **Processor's id**, specify the identifier of the processor that must process documents.
3. Click **Apply**.

## Configuring the XML Element Processor

This processor performs a by-name node (or attribute) selection and push their values as metas. The connector selects nodes and attributes according to an include and an exclude list.

The following rules apply:

- Children of included nodes are pushed as separate metas,
- Children of excluded nodes are completely ignored (as well as their attributes).

1. Expand **XML element processor**, and click **Add item**.
2. Select **Concatenate text element**, to prevent the text to be split when the XML code contains predefined character entities (like `&amp;`, `&gt;`, `&quot;`, etc.).

For example, if you index the tag: `<THEME_FR>Global Procurement & Supply chain</THEME_FR>` without the **Concatenate text element** option, you get several metadata instead of a single one. Instead of a single metadata: `THEME_FR = Global Procurement & Supply chain`, you get 3 metadata:

- `THEME_FR = Global Procurement`
- `THEME_FR = &`
- `THEME_FR = Supply chain`

3. For **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
4. In **Include elements**, specify element mappings:
  - a. **Element**: Name of the XML element to extract. This can also represent an attribute when specifying a string in the form `node@attribute`. For example: `book@title`.
  - b. **Meta**: Name of meta associated with the node content. If you do not specify any meta, the connector uses the name of the element or attribute to infer the meta name. For example, if `Element` is set to `book`, the node text is pushed in the `book` meta, or if `Element` is set to `book@title`, the attribute text is pushed in the `title` meta.
5. **Optionally**, in **Exclude elements**, you can specify a list of node names that must be skipped by the processor.
6. Click **Apply**.

## Configuring the XPath Processor

This processor associates results of XPath selection to metas.

1. Expand **XPath processors**, and click **Add item**.
2. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
3. In **XPath**, specify Xpath mappings:
  - a. **XPath**: XPath expression
  - b. **Meta**: Name of meta associated with the XPath results.
4. Click **Apply**.

## Configuring the XSLT Processor

This processor performs XSL transformations on XML documents and passes the result to its following processors.

1. Expand **XSLT processors**, and click **Add item**.
2. In **Stylesheet**, specify the file path of the stylesheet file.
3. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
4. In **Next processor's id**, specify the list of processor identifiers to run with the transformed XML.
5. Click **Apply**.

## Configuring the Tee Processor

This processor relays processing events to a number of following processors. No data extraction is performed on the XML document.

1. Expand **Tee processors**, and click **Add item**.
2. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
3. In **Following processor ids**, specify the list of the next processor identifiers to run.
4. Click **Apply**.

## Configuring the XML Attach Processor

This processor dumps the current XML document and add it as a part or a meta. You can insert this processor anywhere in the pipeline. It will relay processing events to the following processor.

1. Expand **XML attach processors**, and click **Add item**.
2. In **Next processor's id**, specify the identifier of the next processor in the pipeline.
3. In **Part name**, specify the name of the part that will include the current document.
4. In **Meta name**, specify the name of the meta that will include the current document.
5. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
6. Click **Apply**.

## Configuring the Child Split Processor

This processor splits XML documents in several chunks. Every chunk is then handled as a separate document by a child processor. Every first-level child of the XML document will be extracted.

For example, if you want to index all books from the following file:

```
<books>
  <book><title>title..</title></book>
  <book><title>title..</title></book>
</books>
```

You can use a **Child split processor** to extract every `<book/>` node, and use an **XPathProcessor** to extract the title with the XPath expression `/book/title`.

1. Expand **Child split processors**, and click **Add item**.
2. In **Next processor's id**, specify the identifier of the next processor in the pipeline that must process chunks.
3. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
4. Click **Apply**.

## Configuring the XPath Split Processor

This processor splits XML documents into several chunks according to a small subset of an **XPath** expression.

Your XPath expression must only include direct child selection, for example, `/a/b/c` selects the `<c>` node in `<a><b><c></c></b></a>` expression.

1. Expand **Custom processors**, and click **Add item**.
2. In **XPath**, specify a simple Xpath expression.
3. In **Next processor's id**, specify the identifier of the next processor that must process chunks.

4. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
5. Click **Apply**.

## Configuring the Custom Processor

This processor uses user-supplied Java classes to handle XML documents.

For instructions on how to write such processors, see [Develop Regular and Split Processors](#).

1. Expand **Custom processors**, and click **Add item**.
2. In **Java class**, specify the java class of your processor.
3. In **Processor's id**, specify the identifier of the processor that must process documents. This id must be identical to the **Entry point id**.
4. Optionally, in **Custom parameters**, specify a list of key and values that must be passed to your processor as additional configuration.
5. Click **Apply**.

## Developing a Custom XML Processor

The XML connector is able to run user-supplied code to process XML documents.

### Develop Regular and Split Processors

#### Compile and Deploy a Custom Processor

### Develop Regular and Split Processors

Every XML processor must implement

`com.exalead.papi.connectors.filesystem.xml.processors.common.PAPIXMLConnector`.

However two typical implementations are more reusable: `PAPIDOMProcessor` and `PAPISAXProcessor`.

All processors are given a `com.exalead.papi.helper.Document` attribute called `papiDocument` where processor results must be applied.

### Develop Regular Processors

To develop your own SAX processor, implement a regular SAX `ContentHandler` but extend `PAPISAXProcessor`, instead of `DefaultHandler`.

For example:

```
package com.exalead.papi.connectors.filesystem.xml.processors;
import com.exalead.papi.connectors.filesystem.xml.processors.common.PAPISAXProcessor;
```



```

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
public class CustomSAXProcessor extends PAPISAXProcessor {
@Override
public void startDocument() throws SAXException {
/* TODO */
}
@Override
public void endDocument() throws SAXException {
/* TODO */
}
@Override
public void startElement(String uri, String localName, String name, Attributes
attributes) throws SAXException {
/* TODO */
}
@Override
public void endElement(String arg0, String localName, String arg2) throws
SAXException {
/* TODO */
}
@Override
public void characters(char[] arg0, int arg1, int arg2) throws SAXException {
/* TODO */
}
}

```

If you want to develop a DOM processor, extend `PAPIDOMProcessor`, and implement the `process(org.w3c.dom.Document)` method.

```

package com.exalead.papi.connectors.filesystem.xml.processors;
import com.exalead.papi.connectors.filesystem.xml.processors.common.PAPIDOMProcessor;
import org.w3c.dom.Document;
public class CustomDOMProcessor extends PAPIDOMProcessor {
@Override
public void process(Document domDocument) throws Exception {
/* TODO */
}
}

```

## Develop Split Processors

To develop a splitting processor, extend either `PAPIDOMProcessor` or `PAPISAXProcessor`, and implement `SplitProcessor`.

For a configuration example, see `processors.ChildSplitProcessor`.

## Compile and Deploy a Custom Processor

### Requirements

For requirements on developing custom Java components, see "Customizing CloudView" in the Exalead CloudView Programmer's Guide.

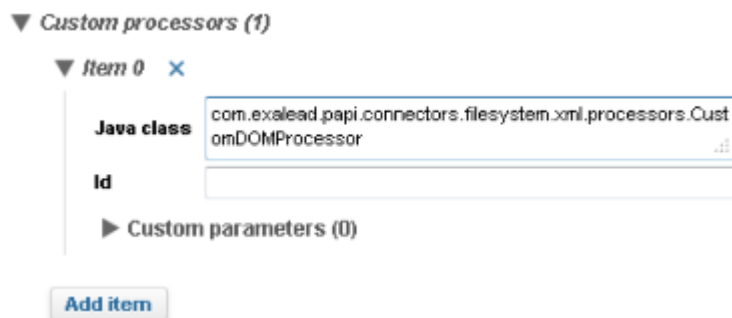
### Compile a Custom Processor

To compile a custom processor, you need the following jars in your build path:

- `<DATADIR>/resource/all-arch/builtin_plugins_deployed/core-java-connectors-plugin/lib/core-java-connectors-plugin.jar`
- `<INSTALLDIR>/sdk/java-customcode/lib/papi-java-client.jar`

### Deploy a Custom Processor

1. Make a JAR containing the processor classes, for example, `custom-processor.jar`.
2. Copy the JAR to: `<DATADIR>/resource/all-arch/builtin_plugins_deployed/core-java-connectors-plugin/lib/`
3. Restart the connector server process.
4. In the Administration Console, add it to the **Custom processors** by specifying its full class name in the **Java class** configuration entry, as shown in the following example.



**Important:** Repeat step 2 after migrating Exalead CloudView to a new version.

## Installing Custom Code

This section describes how to install your custom code in Exalead CloudView to work with the XML connector.

## Requirements

Java JDK (version 1.5 or later) is required for installing Java custom code for this connector.

## Install Custom Code

Before you can use custom code with the XML connector, install it in the Exalead CloudView <DATADIR> as follows.

1. Copy the custom file to:

```
<DATADIR>/resource/all-arch/builtin_plugins_deployed/core-java-connectors-plugin/lib
```

2. Restart the connector server process (if it already exists).
  - In the Administration Console, go to **Home**.
  - In the **Processes** pane, click the restart icon of the **connectors-java0**.