CloudView CV23

Administration

Table of Contents

Business Analytics Server

Business Analytics Server is based on the Exalead CloudView technical product. It allows you to index huge quantities of structured and unstructured data coming from multiple data sources, and present it in an intuitive search interface.

This guide provides instructions to maintain, secure, monitor, and perform day-to-day operations in Exalead CloudView.

Audience

Exalead CloudView and system Administrator.

Further Reading

You might need to refer to the following guides:

Guide	**for more information on**
Installation	installation, deployment, and upgrade
Connectors	standard connectors' configuration.
Configuration	indexing and search concepts, as well as advanced functionality.

What's New?

There are no enhancements in this release.

Product Overview

This section describes key concepts used in Exalead CloudView architecture and administration.

Exalead CloudView Components

Exalead CloudView allows you to index huge quantities of both structured and unstructured data, from multiple data sources, and then present them in an intuitive search interface.

Main Software Interfaces

Exalead CloudView includes the following main software interfaces:

- Administration Console (a.k.a. Admin-UI) is the main user interface for configuring connection to data sources via connectors, indexing options and search processes.
- Mashup UI is the search front end of the Exalead CloudView solution.
- Mashup Builder is the drag-and-drop interface for modifying the front end of the search application (that we call "Mashup UI") and creating custom applications. It lets you use two types of feeds and a limited set of widgets.
- Business Console is a graphical interface for business users to:
    - Control the search results relevance.
    - Manage alerting services.
    - Edit semantic resources used both to enrich documents at indexing time and expand queries at search time.

and

- Monitoring Console is a console to graphically track Exalead CloudView services and resource consumption.
- API Console is a console to edit most configuration files and perform operation commands on Exalead CloudView. It allows you to configure advanced properties that are not accessible in the Administration Console.

For more information on how to display these components, see "Access the interfaces" in the Exalead CloudView Installation Guide.

Exalead CloudView APIs

Exalead CloudView provides the following APIs to allow integration with third-party applications.

- Push API (PAPI) is the public API that allows Exalead CloudView to index data from any source. It supports all operations required to develop new connectors, both managed and unmanaged.
- Mashup API is the API which retrieves the contents of data sources to make them accessible to the data feeds.

  Important: Mashup Builder Premium also uses other APIs for non-Exalead CloudView feeds (for example, FlickR search).

- Search API is the public API for developing third-party search applications. It is the entry point for performing searches on Exalead CloudView.
- Management API (MAMI) is the public API used to configure and manage the Exalead CloudView processes.

For more information, see the Exalead CloudView Programmer's Guide.

TCP Ports

Exalead CloudView uses a continuous range of 100 TCP ports to operate correctly. The default entry points are the following.

Default port	Used by
`<BASEPORT>`	Mashup UI
`<BASEPORT> + 1`	Administration Console, Mashup Builder, Business Console, Monitoring Console, and API Console
`<BASEPORT> + 2`	Push API
`<BASEPORT> + 10`	Search API
`<BASEPORT> + 11`	Gateway and Management APIs

For example, you can access the Administration Console at http://<HOSTNAME>:<BASEPORT +1>/admin.

Supported URLs

The following table describes the URLs supported in the Exalead CloudView configuration.

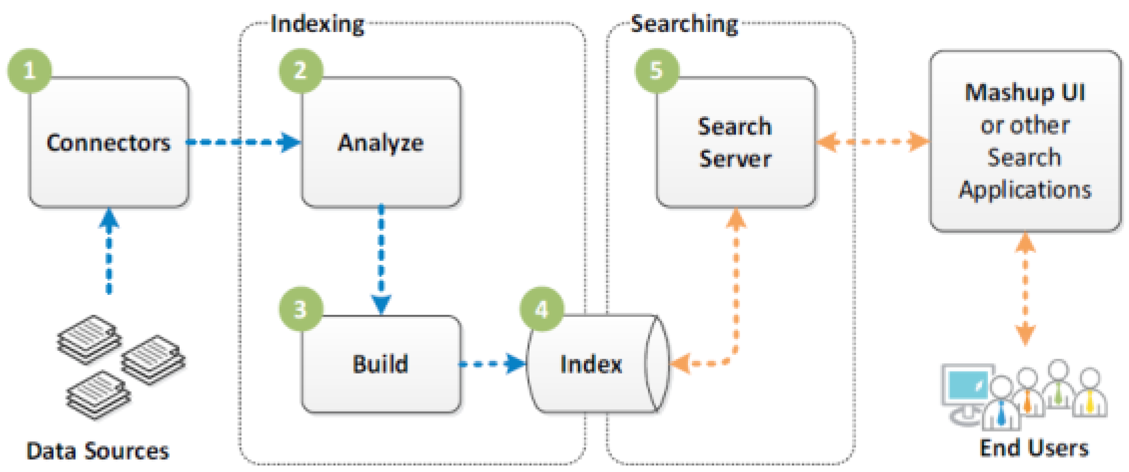
URL	Description	Example
`config:///`	Files located in the applied product configuration.	`config:///2/master/Connectors.xml`
`file:///`	Files located at this file system path. This follows the standard File URI scheme. **Recommendation:** Use the absolute path.	on Windows, `file:///C:/path/to/myfile.txt` on Linux, `file:///data/smith/mytext`
`data:///`	Files located at the root of the `<DATADIR>` of your specific product install, for example, to reference your synonyms resource file.	`data:///synonym/synonym.bin`
`http://`	Standard HTTP URL protocol.	`http://server1:10000/myfiles`
`https://`	Secure HTTPS URL protocol.	`https://server1:10000/myfiles`
`resource:///`	By default, Exalead CloudView looks for resources in `<DATADIR>\resource\all-arch` and then the `<INSTALLDIR>\resource\all-arch`.	`resource:///lemmatizer/LANG`

Understanding the Exalead CloudView Workflow

Understanding how indexing and search work and which process is involved at each step of the Exalead CloudView workflow will allow you to monitor your application more efficiently.

Simplified View

The following diagram summarizes the process to index and then search for documents using Exalead CloudView.



1. Connectors access the data sources (the corpus), convert the files into documents, and then send them to the Indexing Server through the Push API protocol.
2. During the analysis phase, the Indexing Server receives documents and triggers their analysis sequentially, entirely in memory. The analyzers process each document in the job, perform text extraction, semantic processing, custom operations, and mapping.
3. During the build phase, the Indexing Server builds the index entirely in memory (RAM).
4. Once the build is complete for a job, it is imported into the index.
    - It merges the data computed from analysis with the current version of the index.
    - Once this is done, the index is committed and updated, the new documents are available for search.
5. The Search Server interprets and processes the search request (user query). Each user query is processed by the Search Server based on a specific search logic and search target.
6. Search results are displayed either in the Mashup UI (the default search application), or a custom search application created with the Exalead CloudView Search API.

Processes in Detail

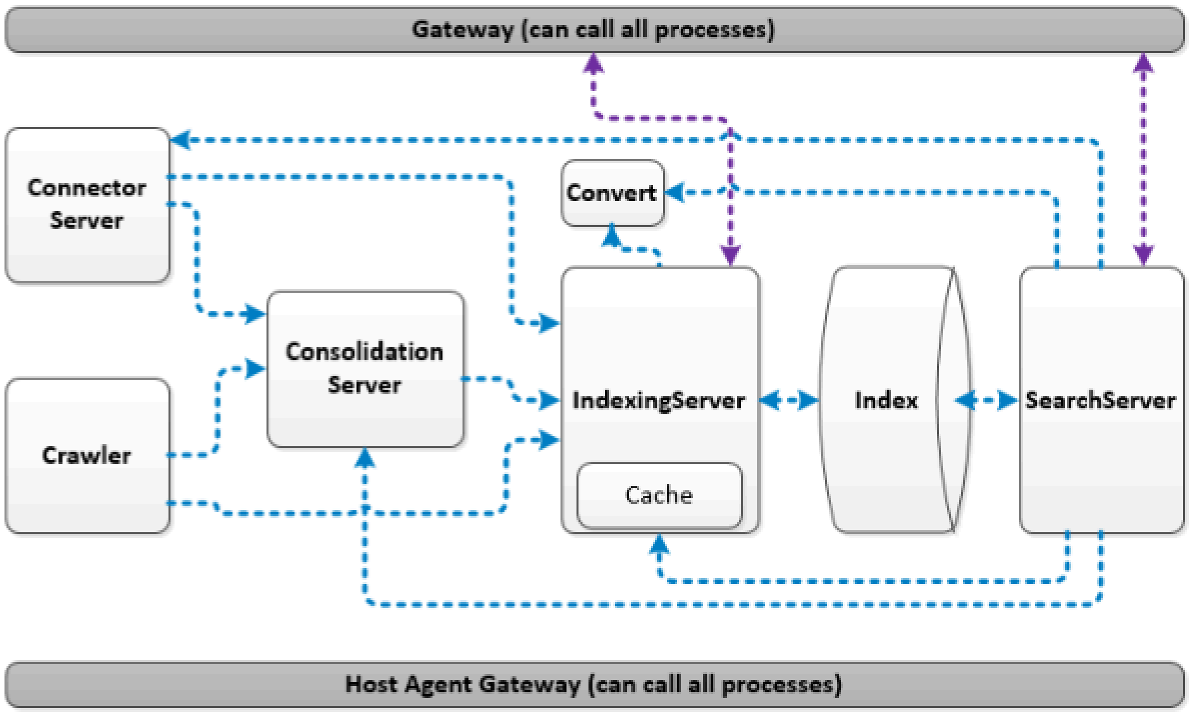
Let us have a closer look at the processes involved at each step of the Exalead CloudView workflow.

Process	**Description**
At connector level	
`connectors`	Manages connectors scan (except custom and crawler connectors).
`crawler`	Manages the crawler connector.
At indexing level	

Process	Description
`consolidationserver`	Manages the consolidation of relational data and incremental updates when changes occur.
`convert`	• Manages document extraction (from text to metas). For example, it allows the product to get textual data instead of binary data. • Builds the preview/ thumbnails for Non-Office documents
`indexingserver`	Manages index building. • PushAPI • Analysis pipeline (document processing, semantic processing, mapping) • Builds the index • Handles replication (if any) • Builds the Search Suggest and the dictionaries
`index6`	Manages indexes and answers search queries.
At search level	
`searchserver`	• Expands the query • Rewrites UQL queries into ELLQL so that they can be interpreted by the product • Polls the index for results • Merges index results and make them understandable • Builds the preview/ thumbnails for Office documents
At configuration level	
`gateway`	Manages Exalead CloudView configuration and resources. • Administration Console • Mashup Builder • Business Console • Performance monitoring • MAMI commands • Alerting

Process	Description
	• Reporting • Apply command • CVdiag
`hostagent`	• Manages the different processes and their communications. It is the parent process that also handles the start/ stop of all processes. • Manages the configuration versions. • Manages the monitoring of process statuses.
`master`	• Handles the MAMI commands that are not handled by the gateway, that is, MAMI Crawl and MAMI Deploy • Manages replication between master and slaves

The following diagram shows all possible interactions between the various processes. These will actually depend on your deployment scenario.



Focus on Indexing

The Exalead CloudView index is generational. When indexing documents, they are divided into batches known as jobs.

Each time a job is indexed, it creates a new generation of the index. Exalead CloudView stores this new generation of the index in a data structure called a slot. Each new slot is appended to the original index. Once the latest generation is committed to the index, the index replicas are updated.

At search time, Exalead CloudView searches in all these slots of all the index replicas, and merges the results to return the final result set. From time to time, slots are compacted to create an index with fewer slots, for more efficient searching.

Build Groups

This section describes how to configure and use a build group.

What Is a Build Group?

The component that takes care of indexing is known as a **build group**.

How Does It Work?

The **build group** is responsible for:

- Pushing and analyzing documents.
- Creating the source index from which all the index replicas in your deployment are based.

Each build group contains:

- An **indexing server**, responsible for pushing and analyzing documents, and then importing the information into the next generation of the index.
- A specific configuration for the **data model** that defines the document processing and mapping done during analysis.
- A specific configuration for **indexing** that defines:
    - How many analyzers (threads) to use for analysis.
    - How often to compact the index.
    - The conditions to trigger a commit of documents on disk to the index.

- An index.

Note: When the indexing server makes a commit, all build group components (DIH, thumbnails, document cache, connector checkpoints, index inverted lists, and index dictionaries) are saved to disk atomically.

By default, Exalead CloudView installs with:

- A default build group, `bg0`.
- A default data model, `default_model`.
- A default indexing configuration, `ic0_standard`.

When to Use Several Build Groups?

You can deploy Exalead CloudView with one or more build groups.

You can:

- Add more build groups to an existing host. For example, when you need to do one of the following:
    - Accommodate multiple corpuses with different indexing schedules, or different data models.
    - Index your corpus on a rotating basis. For example, if `it1` indexes on week 1, then `it2` indexes on week 2 while `it1` is cleared.
- Add more build groups to a new host to increase corpus volume and indexing throughput.

For more information, see "Add a new build group" in the Exalead CloudView Installation Guide.

Configure a Build Group

This procedure describes how to configure an index build group.

1. In the Administration Console, go to **Deployment > Build Groups**.
2. Select your build group (by default, **bg0**).
3. For **Data Model**, you can change which metas are indexed, how they are organized into classes, or how they are processed in the analysis pipeline by selecting another model.

   For more information, see the "About the Data Model" section in the Exalead CloudView Configuration Guide.
4. For **Indexing config**, you can configure how many threads to use for the analysis, how often to compact the index, and how often to create new index generations.

   For more information, see Configure Indexing.
5. To enable document caching by connectors, select **Document cache**.

For more information, see Document Cache.

6. Click **Apply**.

Configure Indexing

This section describes how to configure indexing to tune both index-time and search-time performance.

You can configure indexing by selecting options to:

Analyze

The critical element in analysis is the number of threads to use, to maximize your indexing machine's CPU.

Most deployments have a dedicated machine for indexing building. An index constitutes of slices. You normally want to set up your analysis to use one CPU per index slice. This is to maximize performance at both index-time and search-time.

Best Practice

Analysis can maximize CPU use when there is a dedicated machine for index building. The best option is one thread per slice. When you have a lot of CPU but a relatively slow hard disk drive, you can increase this to two threads per slice.

An exception to this best practice is when you have a document processor that calls a web service. While waiting for the response from the web service, Exalead CloudView does not consume CPU. In this case, put more threads than there are CPUs to compensate for the periods when the CPU is not working at all.

Why 1 CPU Per Slice?

Say that we have an index with 4 slices. At search time, when a user sends a query for processing, the search server sends the query to each slice. Query evaluation is mono-threaded in each slice, which means one CPU for each slice, and maximized query performance.

At index building time, you must import the new index generation into each slice.

Recommendation: Have at least one CPU per thread for indexing time.

Compact

Compacting is the merging of multiple indexing generations, known as slots. Doing this regularly helps to keep good search performance.

Note: Changes made in **Compact** do not require re-indexing. A full compact is enough to clean index slots.

Regular Vs Full Compact

There are two types of compacting you must configure for Exalead CloudView:

- "Regular" compacts: these are for daily housekeeping on the index. They regularly merge slots for more efficient use of index space.
- Full compacts: by contrast, these are for spring cleaning on the index. They are triggered once regular compacting has lost its effectiveness.

"Regular" Compact

Each analyzer imports the new index generation into each slice. To ensure consistency, Exalead CloudView creates new files, or slots, for each generation. Since more slots slow search performance, you do not want to add new slots indefinitely. You sometimes need to merge these slots by compacting the index.

In general, small slots are faster to compact, while large slots maintain a good search performance. See the table below for a description of the available compact policies.

Compact policy	Description and options
Number of slots (default)	Compacts as soon as there are **No. slots** slots. This is a pyramidal system. It leads to frequent compacting of small slots and less frequent compacting of large slots. • **No. slots**: Number of slots with the same number of index generations to trigger a compact. Default is 4. • **Max slot size (MB)**: Once a slot reaches this size, it can never be compacted again unless you activate a full compact policy. Default is 1000.
Latency reduction	Compact policy designed to improve realtime indexing performance. Small slots (small size on disk) provide fast compacts, while large slots (large size on disk) maintain a good search performance. Whenever an index generation is created, this compact policy sorts the slots by size: with **No. large slots** large slots and **Max small slots** small slots. Use this mode when most of your index imports are for incremental changes, which typically create small slots.

Compact policy	Description and options
	• **No. large slots**: Keeps at least N large slots in the index. Default is 10. • **Max small slots**: Keeps no more than N small slots. Exceeding N small slots triggers a compact. Default is 20.
Slots size	Compact policy based on size that produces slots with similar sizes. • **Target size for compaction (MB)**: Slots are compacted until they reach this size. They are no longer compacted afterward, except if you run a full compact operation. Default is 200. • **Min size for compaction (MB)**: Minimum size for a slot to be compacted. Default is 50. • **Min. slots**: Minimum number of slots to trigger a compact.
No compact	Compact policy that does not run compact operations, and fill the smallest slot at each import. Use for initial indexing, when all you are doing is importing. Follow this with a **Full compact** (see below).

Full Compact

A full compact is like spring cleaning for your index.

To ensure good index latency, Exalead CloudView creates lots of small slots, one for each generation of the index. For better search latency, every so often these slots are compacted into a large slot. Later on, once you have lots of larger slots, these too get compacted. For the sake of clarity, let us call this a 'regular' compact.

Once a slot reaches 1 GB, however, regular compaction stops. This is a safeguard put in place to ensure that regular compaction does not impact other Exalead CloudView operations. This means that over time, your index becomes full of 1 GB slots. This is particularly wasteful when you are indexing the same docs repeatedly, since each slot contains new versions of the same documents.

This is when it is time to do a full compact, which takes all these 1 GB slots and merge them into a single slot.

By default, a full compact is triggered using the size of the largest slot in your index as the threshold. Once the size of the rest of your index (excluding the largest slot) exceeds the size of the largest slot, it triggers a full compact.

Note: You can also trigger a full compact for a given build group directly, from **Administration Console > Home** using **Full compact**.

Important: To perform a full compact, you need free disk space equal to the size of your index. Once fully compacted, though, this space is no longer required. If the index is made of multiple slices, you can limit this extra disk consumption by limiting the number of slices compacting at the same time. Specify this limit in `Indexing.xml` under `CompactPolicies`, using the `maxParallelFullCompacts` parameter.

Full compact policy	Description
Size	This full compact policy is launched when the cumulated size of small slots exceeds N percent of the largest slot. **Recommendation:** Set the Min slots option so that full compact operations will not be launched too frequently, as it is costly in disk consumption. - **Percentage**: Minimum percentage to start a full compact. It compacts all slots into a single one whenever the tail of small slots exceeds a specific percentage of the largest slot. - **Min slots**: Minimum number of slots to trigger a full compact. Default is 2.
Number of slots	This full compact policy applies to Compacts based on Number of slots. Since the pyramidal system tends to compact large slots less frequently, this policy allows you to define the max arity of long tails before triggering a full compact. **Max arity**: Whenever the long tail total arity reaches this Max arity, a full compact is launched. The long tails are the slots whose span has an arity inferior to this parameter. Default is 256.
No full compact (default)	Disables full compact operations. **Warning:** Do not use it after performing the first indexing operation.

Schedule Full Compacts

During full compacts, index queries may be slower than usual if the service index is on the same machine as the `indexingservice` process. To mitigate this, schedule full compacts when there is less traffic on the system. Depending on the update volume, you may want to trigger full compacts every night, or once a week.

You can do this in `Scheduling.xml`.

For example, to trigger a full compact every night at 01:00:

```
<master:SchedulingConfig version="1381920589000" xmlns:bee="exa:exa.bee"
xmlns:cdesc="exa:com.exalead.mercury.component.config.descriptor"
xmlns:secs="exa:com.exalead.security.sources.common"
xmlns:config="exa:exa.bee.config"
xmlns:master="exa:com.exalead.mercury.mami.master.v10">
  <master:JobConfigGroup name="full_compact">
    <master:DispatchJobConfig name="launch_full_compact">
      <bee:DispatchMessage messageName="fullCompactIndex"
      serviceName="/mami/indexing">
        <bee:messageContent>
          <bee:KeyValue key="buildGroup" value="bg0" />
        </bee:messageContent>
      </bee:DispatchMessage>
    </master:DispatchJobConfig>
  </master:JobConfigGroup>
  <master:TriggerConfigGroup name="full_compact">
    <!-- schedule full compact -->
    <master:CronTriggerConfig name="launch_full_compact" startTime="0"
    endTime="0" jobGroupName="full_compact" jobName="launch_full_compact"
    cronExpression="00 00 1 * * ?"/>
  </master:TriggerConfigGroup>
</master:SchedulingConfig>
```

Synchronous Option

By default, compacting is asynchronous. When importing the latest generation of the index, Exalead CloudView creates the slot. While replicating this slot to all slices, Exalead CloudView can start a compact, but does not wait for this compact to fully replicate before responding to the user queries.

With synchronous compacting, Exalead CloudView ensures that compaction is fully replicated before starting an import. This prevents machines from being overloaded with multiple compacting or importing jobs.

Commit

Exalead CloudView indexes documents on the fly, all in memory. As soon as a connector pushes documents to Exalead CloudView, their data processing analysis begins. During the analysis, if a threshold is reached, a commit of processed documents is made to the index (on disk). The commit is what creates a new generation (slot) on the index.

The commit thresholds are commit conditions. You can define them to occur:

- At regular intervals (periodic condition)
- After the process of N MB (size-based condition)
- After the process of N tasks (documents) (number of tasks condition)

- After N seconds of inactivity (inactivity condition)

Note: You can also explicitly choose to commit from the **Home** page > **Indexing** section> **Force commit**.

By minimizing writing to disk during the analysis phase, indexing is significantly faster and the end is reduced index latency. In other words, your result users can search these updated documents sooner.

Keep in mind, though, that while scanning (pushing) documents on a data source, connectors also make commits. For example, at the end of every scan, a managed connector performs a commit, and consequently, an import to the index. Importing too frequently could negate the advantages of RAM-based analysis.

Commit based on...	Description
Max. RAM threshold	- **Enabled** (default option): Commits when the RAM size reaches the **Threshold value** specified (by default, 2048 MB). - **Auto**: Commits when the RAM size reaches 2048 MB. When reaching the RAM value specified, analysis stops and analyzed documents are written to the index. Then analysis starts again.
Inactivity	Commits by inactivity when: - there is no new data for the specified time period (`n` seconds) - and at least `n` tasks have been analyzed
No. of tasks	Commits after `n` tasks have been issued.
Elapsed time	Commits every N second after the first push order launched after the last commit.
Size	Commits when the total number of documents to be processed reaches `n` MB.

Document Cache

You can use the document cache when you have a slow connector, or want to speed up indexing throughput.

The document cache stores documents pushed by a connector, before being processed.

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

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 you add a tag, 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 specify its maximum and minimum size.
3. Apply your changes.

Change the Location of the Document Cache on the File System

By default you can find the document cache 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 status, move the original document cache directory to the new location specified in Document Cache.

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.

When to Reindex?

The table below indicates when to reindex according to the sections modified in the Administration Console.

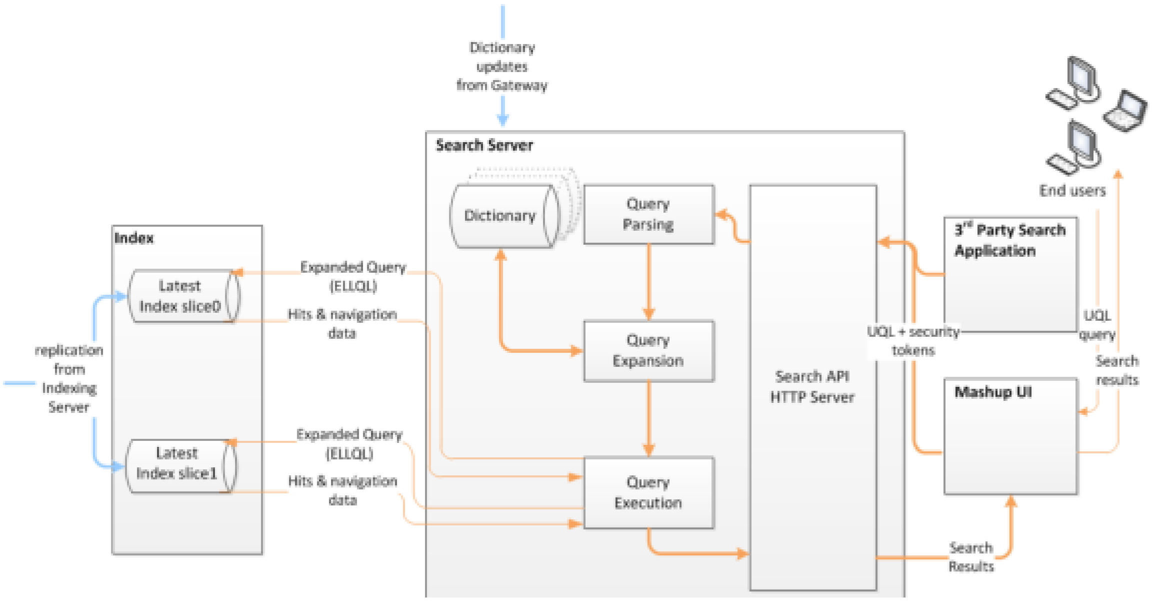
Modifications made on...	**Needs reindexing?**
Index	
Connectors	**Yes**
Data Processing	**Yes**
Data Model	**Yes** **Note:** When you add a property to the data model, you only need to reindex to search and retrieve this new property.
Linguistics	**Yes**

Modifications made on...	Needs reindexing?
Tuning	No Changes made in **Analyze** and **Commit** require a restart of Exalead CloudView, or at least the indexing server process, to be taken into account. Changes made in **Compact**, do not require reindexing, a full compact is enough to clean index slots.
Search	
Search Logics	No
Security Sources	No
Search API	No
Suggest	No
Reporting	No
Deployment	
Roles	No
Build Groups	**Yes**
Plug-ins	No
Resources	No

Focus on Search

Search is triggered when end users (or a third-party application) submit a query to Exalead CloudView.

The following diagram shows how Exalead CloudView parses and expands queries before searching for matches on the index replicas.



1. End users perform a search query using either the Mashup UI or the Mashup API (via 3rd party search applications).
2. The Search API received the query.
3. The query is parsed (that is, words and separators used in the query are checked) and expanded using the dictionary. For example, synonyms are added to the query.
4. The expanded query is executed. It is broken down into a more granular query language known as ELLQL (EXALEAD Lower-level Query Language) so the index slices can understand it.
5. Most relevant matches are searched for in all index slices. Navigation data (for example, facet) are generated.
6. Hits from each slice are merged in the search server. Search results are returned to the user.

Maintaining Your Installation

This section describes how to perform common maintenance operations on your Exalead CloudView installation.

Administration Tools

This section describes common administration tools.

Overview

You can manage the Exalead CloudView platform administration with the following tools.

Feature	Description
Administration Console	Use this user interface to configure and control most administrative operations. See Display System Information Via the Administration Console.
API Console	Provides a complete interface for Exalead CloudView configuration and administration. Use this API to control (configure, operate, inspect) the product from an external program. See Perform Advanced Operations with the API Console.
`cvinit` tool	Use this command-line tool for administrative tasks on your Exalead CloudView instance (for example, start/stop).

Feature	Description
	See cvinit.
`cvcommand` tool	Use this command-line tool from any location to send commands to any running Exalead CloudView product. The `<DATADIR>\bin\cvcmd` wrapper is also available for quick command service scripting. See cvcommand and cvcmd.
`cvcmd` tool	Use this cvcommand wrapper for quick `cvcommand` service scripting only. See cvcommand and cvcmd.
`cvdebug` tool	Use this command-line tool to perform debugging operations. See cvdebug.

Display System Information Via the Administration Console

The **Home** page of the Administration Console (http://<HOSTNAME>:<BASEPORT+1>/admin) is the entry point for status information and common indexing operations.

Connectors

The following actions can be performed in the **Home > Connectors** section:

Action	Description
Clear documents	Deletes all connector documents from the system and resets the connector state.
Scan	Performs a full scan for a specific connector. This forces an immediate commit and updates the documents in the index.
Abort scan	Stops the scan for a specific connector. **Note:** It does not delete the documents already processed.

Indexing

The following actions can be performed in the **Home > Indexing** section:

Action	Description
Clear	Opens a pop-up window to delete specific content:

Action	Description
	• **Master index data**: Deletes the index data associated with this Build Group. **Note:** It does not delete the dictionary content. • **Replica index data**: Deletes the replicated index data. • **Document cache**: Deletes the Cache content for the selected Build Group. Option available only if Document Cache is enabled in **Deployment > Build Groups**. • **Precomputed thumbnails**: Deletes the Thumbnails content for the selected Build Group. Option available only if thumbnail generation is enabled. **Note: Thumbnails details** do not get reset until you restart the **indexing server** process. • **Dictionary data for all Build Groups**: Deletes the dictionary content for all Build Groups. **Recommendation:** Clear all Build Groups after this operation.
Force commit	Takes all documents processed in RAM and saves them on disk in the index.
Full compact	Triggers a full compact operation for a given build group.
Replication > Clear	Clears the replica for a given slice.
Replication > Detach	Detaches the replica for a given slice (available only if you are not using the index builder directory).

The **Build group status** section in **Home > Indexing** gives the details of a specific build group and of the associated replication process.

Processes

The **Home > Processes** section provides status information for all the processes of the specific host.

For each process, you can:

- Jump to its related log by clicking its link.
- Manage the process using the icons in the **Action** column:

- start
- restart
- stop
- kill

Perform Advanced Operations with the API Console

Why Use It?

The API Console exposes administration, inspection, and operations for all the management objects. These objects give complete control over the product's configuration parameters, as well as status information for the platform's components.

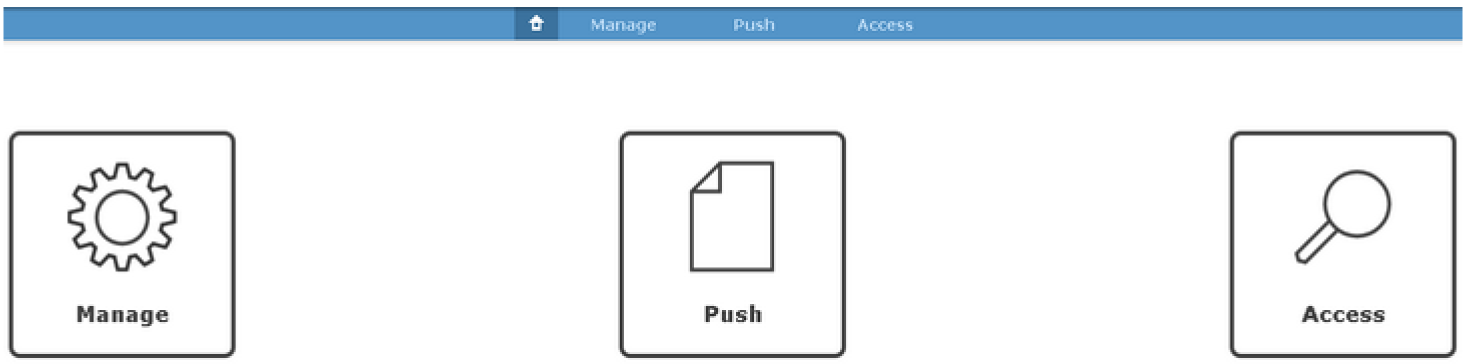
The API Console provides access to Exalead CloudView APIs (known as MAMI), allowing you to:

- Edit most configuration files and apply changes. The exceptions are the configurations for Mashup Builder, Business Console (as well as Content Recommender component of the Business Console, if installed).
- Push simple documents to test your searches.
- Perform XML searches.
- Perform all operation and inspection commands on Exalead CloudView.

Where to Access It?

The API Console is available at:

http://<HOSTNAME>:<BASEPORT+1>/api-ui



Select an API Console menu, that is **Manage**, **Push**, or **Access**.

Error Handling

In the event of an error, for example, `REQUEST TO SERVER FAILED!`, the API Console provides a log for complete details.

If an error occurs, press `SHIFT + H`. To pause the log, press `SHIFT + Z`.

Manage a Configuration

The API Console gives you access to configuration and monitoring for all Exalead CloudView Management objects via the **Manage** menu. Each service has procedures allowing you to:

- Configure
- Operate
- Inspect

This service	Manages Exalead CloudView's...
`adminui`	Administration Console
`alerting`	Real-time alerting
`connect`	Data source connectors (except HTML)
`consolidation`	Consolidation Server process
`convert`	Document conversion process
`crawl`	HTML connector
`datamodel`	Data model classes and properties
`deploy`	Server deployment
`fetch`	HTTP fetchers
`indexing`	Index build process.
`linguistic`	Linguistic configuration
`master`	Deployment, including applying changes, job scheduling, and security
`resource manager`	Resources configuration
`search`	Search API
`thumbnails`	Thumbnails generator

Access the Search Engine

Once the platform is configured, and the index is built, you can access the search engine via the **Access** menu. You can view the search results either in XML or in Text format.

To test the search engine and get the results as text:

1. Click **text** to access this search engine.
2. Enter the search terms in the query box.
3. From the lists, select the search application and the search page you require.
4. Add security tokens to the search if required.
5. Click **Search**.

This displays your search results in text format.

To test the search engine and get the results as XML:

1. Click **xml** to access this search engine.
2. Enter the search terms in the query box.
3. From the lists, select the search application and the search page you require.
4. Add security tokens to the search if required.
5. Click **Search**.

This displays a structured and color-coded view for the results of your query. The syntax highlight displays for markup languages only.

Push Documents

In the API Console you can push individual documents to the index using the **Push** interface. A document can have multiple metas and parts, however it must have a unique identifier and time stamp.

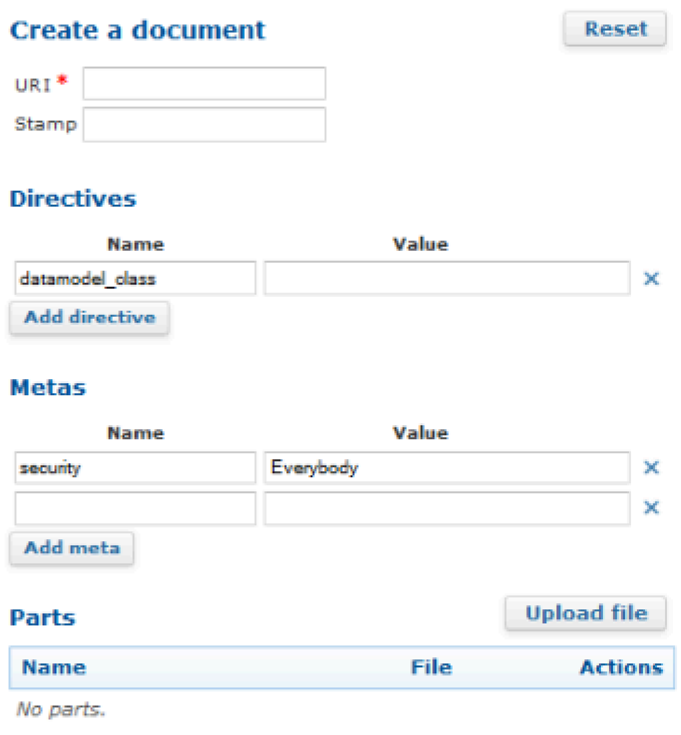
Access the Document Collection

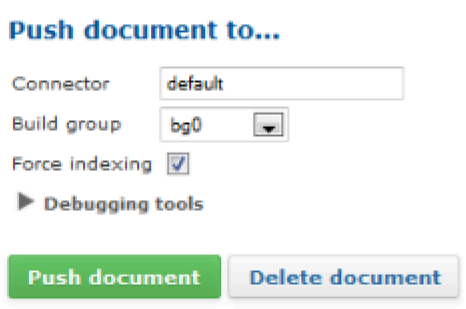
1. Open a browser and enter the Exalead CloudView API Console URL:

   http://<HOSTNAME>:<BASEPORT+1>/api-ui

2. Select **Push**.

   The interface to push documents displays.





Push Documents

Only the URI and the part name are required fields.

1. Enter the unique document identifier in **URI** and the time **stamp** (optional).

   For example, `myuri` and `2018/03/28-08:00:00`.

2. In the **Metas** section, specify the meta details for the document in **meta name** and **value**.

   For example, enter `department` and `marketing`.

3. Click **Add meta** to create new metas and repeat step 2. Click **X** to remove any row.
4. Click **Upload file** to select your document.

   Your document's name is `master`. Click this name to display:

   ◦ The **filename**. For example, `doc`.

   ◦ The **encoding** type. For example, `UTF-8`.

   ◦ The **mimeHint**. For example, `text/richtext`.

5. Select the connector and build group to which you need to push the document.
6. Click **Push document**. The document is pushed into the index with the associated metas.

   Note: You can also display checkpoints (if any) and document status.

Get Started with Command-Line Interfaces

cvinit

Start/Stop Exalead CloudView

To perform basic start/stop/restart operations on your Exalead CloudView instance, go to the `<DATADIR>/bin` directory and run the command-line scripts as described below:

To...	Use
Start Exalead CloudView	`cvinit.[bat|sh] start`
Stop Exalead CloudView	`cvinit.[bat|sh] stop`
Restart Exalead CloudView	`cvinit.[bat|sh] restart`
Get the status of the Exalead CloudView processes.	`cvinit.[bat|sh] status`
Force Exalead CloudView to stop.	`cvinit.[bat|sh] kill`

Note: In Windows environments, you can use the Microsoft Management Console to start and stop the registered service or use the command line. For example, to start the product: `net start "Exalead CloudView Search - cvdefault"`.

Advanced Operations

To perform advanced operations on your instance, go to the `<DATADIR>/bin` directory and run the command-line scripts as described below:

To...	Use
Wait until the start of all Exalead CloudView processes	`cvinit.[bat|sh] wait-started`
Check that indexes are up on a host	`cvinit.[bat|sh] check-indexes-up`
Detach a Exalead CloudView host from the	`cvinit.[bat|sh] detach` OR use method:

To...	Use
primary server (to run on secondary server) For more information, see Detach Indexes from the Replication Process.	`detachHost` in API Console
Attach a Exalead CloudView host to the primary server (to run on secondary server) For more information, see Detach Indexes from the Replication Process.	`cvinit.[bat|sh] attach` OR use method: `attachHost` in API Console
Force 'isAlive' health monitors on a host as down For more information, see Safely Stop Indexes.	`cvinit.[bat|sh] set-not-alive`
Force 'isAlive' health monitors on a host as up For more information, see Safely Stop Indexes.	`cvinit.[bat|sh] set-alive`
Return whether the 'isAlive' health monitors have been forced as down For more information, see Safely Stop Indexes.	`cvinit.[bat|sh] is-alive`

cvcommand and cvcmd

What Is the Difference

Tool	Purpose	Example
`cvcommand`	Sends HTTP requests to the product gateway.	To display a build group status, run:

Tool	Purpose	Example
	The service can be any valid service mounted on the gateway.	`cvcommand myhost:10011 /mami/indexing getBuildGroupStatus buildGroup=bg0` or go to: `http://myhost:10011/mami/indexing/getBuildGroupStatus?buildGroup=bg0`
`cvcmd`	Sends commands using the `command` service only.	To test a configuration, run: `cvcmd testConfig` or `cvcommand myhost:10011 command testConfig`

Use cvcommand

The `cvcommand` is a command-line tool located in the `<DATADIR>\bin` directory. Its syntax is the following:

```
cvcommand [options] gw service method [args]
```

Where:

- `gw`: gateway address (`http://path_to_gateway` or `host:port` or `:port`)
- `service`: name of the service
- `method`: name of the method
- `args`: argument list in the form `name=value`

Options:

- `help|-h`: displays the help
- `timeout|-t` seconds: number of seconds to wait for an answer

Error codes:

`cvcommand` returns a 0 error code if operation is successful. You can get the error code:

- On UNIX, using "`echo $?`".
- On MS Windows, using `echo %ERRORLEVEL%`.

Common Scripts

To...	Use...
Display help messages	`cvcommand -h`

To...	Use...
	or `cvcmd help`

Display status process list	`cvcommand host:10011 command status`
	or
	`cvcmd status`

Run `cvcommand` from another machine	`cvcommand myhost:10011 command status`
Run `cvcommand` from another machine, which has only access to the gateway through an http proxy (`http://gateway.mycompany/gateway`)	`cvcommand GATEWAYURL/gateway command status`
Test the configuration before applying	`cvcommand myhost:10011 command testConfig`
Apply the configuration	`cvcommand myhost:10011 command applyConfig`
Specify logging level to `DEBUG`	`cvcommand myhost:10011 command setLoggingLevel level=debug`
Get the exhaustive list of available internal services	`cvcommand myhost:10011 directory getServices`
Get the list of service descriptions	`cvcommand myhost:10011 directory getSchemas`
Retrieve errors logged in `log.log` using filters (date, component, level)	`cvcommand myhost:10011 /mami/master getGlobalLogEntryList (startDate, endDate, maxEntries, codeFilter, componentFilter, levelFilter)`
Display build group status (analysis, import, indexing)	`cvcommand myhost:10011 /mami/indexing getBuildGroupStatus buildGroup=mybuildgroup`
Enable, for a given index build group: - Analysis - Import	`cvcommand myhost:10011 /mami/indexing enableAnalysis buildGroup=mybuildgroup` `cvcommand myhost:10011 /mami/indexing enableAnalysis buildGroup=mybuildgroup`

To...	Use...

Disable, for a given index build group:
- Analysis
- Import
	`cvcommand myhost:10011 /mami/indexing disableAnalysis buildGroup=mybuildgroup`
	`cvcommand myhost:10011 /mami/indexing disableImport buildGroup=mybuildgroup`

Display the list of analysis configuration	`cvcommand myhost:10011 /mami/indexing getAnalysisConfigList`
Clear an index build group	`cvcommand myhost:10011 /mami/indexing clearBuildGroup buildGroup=mybuildgroup`

cvdebug

cvdebug is a command-line tool, which allows you to perform advanced analysis and indexing debug. The `cvdebug` tool is in the `<DATADIR>\bin` directory.

To dump an index:

```
cvdebug index print slice=1 buildGroup=mybuildgroup > /path_to_output_file
```

To submit a document to test processors in the pipeline:

```
cvdebug analysis analyze path=<PATH_TO_DOCUMENT>
```

For more information, see cvdebug.

Managing Configurations

Exalead CloudView has a versioned configuration. When you save a modification in the Administration Console, the new configuration does apply immediately to the running instance. Instead, you can find it in the configuration store. When you apply the configuration, it applies to all running components.

You can make configuration changes using:

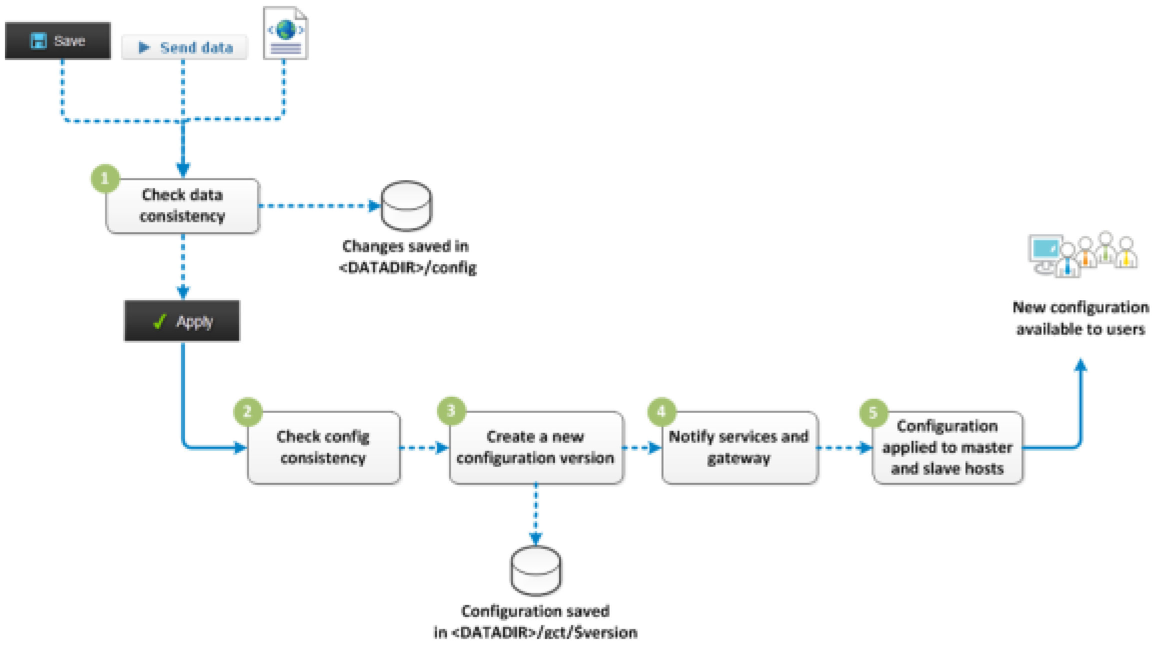
- The Administration Console web interface (http://<HOSTNAME>:<BASEPORT+1>/admin)
- Manually via:
    - the API Console (http://<HOSTNAME>:<BASEPORT+1>/api-ui)
    - the XML configuration files (`<DATADIR>/config` directory)

Using either of these tools, changes are submitted to master and slave hosts once you apply changes to the existing configuration.

Note: Very few configuration changes need a global restart. If required, you receive a request to do so.

How Saving and Applying Changes Work

The diagram below describes what happens when clicking **Save** or **Apply**.


Save
Send data
1
Check data consistency
Changes saved in <DATADIR>/config
Apply
2
Check config consistency
3
Create a new configuration version
4
Notify services and gateway
5
Configuration applied to master and slave hosts
New configuration available to users
Configuration saved in <DATADIR>/gct/$version

1. You modify the configuration either in the Administration Console, in the API Console, or directly in `.XML` files. Exalead CloudView checks the consistency of modified data. You can find the modifications in `<DATADIR>/config`.
2. Exalead CloudView checks the consistency of modified data with other configuration parameters.

   Note: You can also click **Apply** directly. In that case, data and configuration consistency are checked simultaneously.

3. A new version of the configuration is created in `<DATADIR>/gct`. It is saved for future rollbacks, if required.
4. Services and gateway are notified of the new configuration version.
5. Configuration applies to master and hosts:
   - If required, modified processes are restarted.

◦ If required, new processes are started.

If a host is down during this process, the latest configuration version applies at startup.

Note: For the search server, all changes made to the search logic are processed as online updates. Queries are paused and processed again once the new configuration applies to the search server.

Compare Configuration Versions

You can select two configuration versions and compare their XML configuration files through theAdministration Console. This can be helpful to roll back to a previous configuration version.

1. From the top navigation bar of the Administration Console, click the down arrow next to **Apply**.
2. Select **Show all previous versions**.
3. Select two configuration versions and click **Compare versions**.

   The **Changed files** window opens, highlighting the differences between the two selected versions.

Roll Back to a Previous Configuration

You can roll back to a previous configuration version through the Administration Console. This replaces the entire configuration store with the previous configuration.

1. From the top navigation bar of the Administration Console, click the down arrow next to **Apply**.
2. Select either the latest version or display all previous versions to compare configurations and click **Rollback**.

Apply Changes When Exalead CloudView Has Stopped

In rare cases, a severe configuration conflict can prevent Exalead CloudView from starting. As a result, the standard tools to apply changes are unavailable.

For this scenario, use the `buildgct` emergency configuration application tool.

1. Stop Exalead CloudView on all servers.
2. Edit the configuration files to fix the error.
3. Run `<DATADIR>/bin/buildgct`.
4. If it fails, continue editing.
5. Restart the master host.
6. Wait for the master host to fully start.
7. Restart the other hosts, if any.

Important: Do not use `buildgct` except for the previous procedure, especially for multihost deployments. The `buildgct` tool does not contact the other hosts to synchronize the configurations, so it can cause severe inconsistencies in the product configuration.

Controlling Disk Space

If you encounter disk space issues, you can:

Reduce Disk Space

To reduce disk space, you can safely remove the content of the following directories:

- `<DATADIR>/tmp`. You can remove it when the product is stopped.
- `<DATADIR>/run`: contains log files. To free up space permanently in this directory, consider using log purge and rotation.
- `<DATADIR>/reporting_store`: contains SQLite databases (queries, analysis/import/compact events, restarts)
- `<DATADIR>/**/*.tmp.*`: temporary files. You can remove it when the product is stopped.

You can also delete old configuration versions stored in `<DATADIR>/gct` with the API Console using the `deleteVersionsBefore` method.

Maximize Disk Space

You can maximize disk space either by compacting your index or by disabling the document cache.

View `<DATADIR>/build/index-unit/<buildgroupname>/` and see which subfolder is taking up the space:

- If the **index** directory is taking a lot of space, it may be that it is not compacted. You can perform a full compact, but keep in mind you need twice the size of your index available on disk to perform this. Otherwise, perform your compact on a slice by slice basis.
- If the **cache** directory is taking a lot of space, you can deactivate the document cache option either at build group or at connector level.

Performing Maintenance Without Service Interruption

For maintenance purposes, you may need to stop your master production index and redirect queries to another host or/and detach indexes from the replication process.

Safely Stop Indexes

When stopping a production index, you want to make sure to process all user queries beforehand.

You can force a "not alive" status for the index, so the load balancer can redirect queries to an available index. Meanwhile, the index processes any queries submitted before you forced the status to "not alive."

Once the index has processed the remaining queries, you can safely stop the index.

Specify the Full Instance to "Not Alive"

In your `<DATADIR>/bin` directory, use the `cvinit` command with one of the following arguments:

1. `set-not-alive`: Forces all 'isAlive' health monitors on this host as down. Use with a load balancer. For example:
   ```
   cvinit.[bat|sh] set-not-alive
   ```
2. `set-alive`: Forces all 'isAlive' health monitors on this host as up. Use with a load balancer.

Specify the Mashup UI to "Not Alive"

Send an HTTP POST request to the following addresses:

1. To specify to not alive: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/isAlive/setDown`
2. To specify it back to alive: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/isAlive/setUp`

Specify the Search API to "Not Alive"

Send an HTTP POST request to the following addresses:

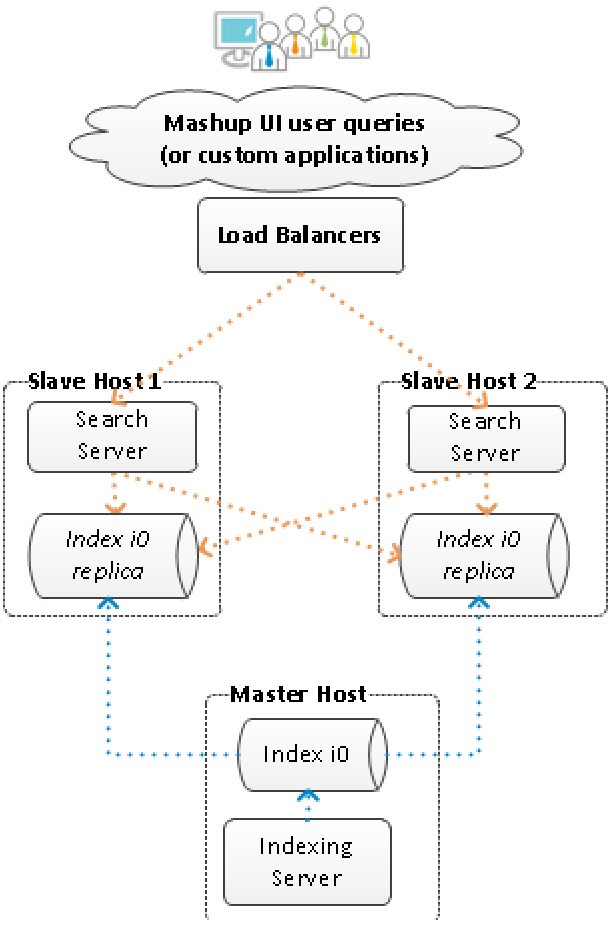
1. To specify to not alive: `http://<HOSTNAME>:<BASEPORT+10>/search-api/isAlive/setDown`
2. To specify it back to alive: `http://<HOSTNAME>:<BASEPORT+10>/search-api/isAlive/setUp`

Detach Indexes from the Replication Process

The goal of replication is to distribute copies of the index generated on a primary server to one or several secondary servers hosts having the 'index' role. The primary server is then dedicated to index operations while search queries are sent to secondary servers hosting index replicas.

How Does It Work

Indexes are usually divided into slices. You can define which slices to replicate on each host when defining host roles. For more information, see "Configure roles" in the Exalead CloudView Installation Guide.

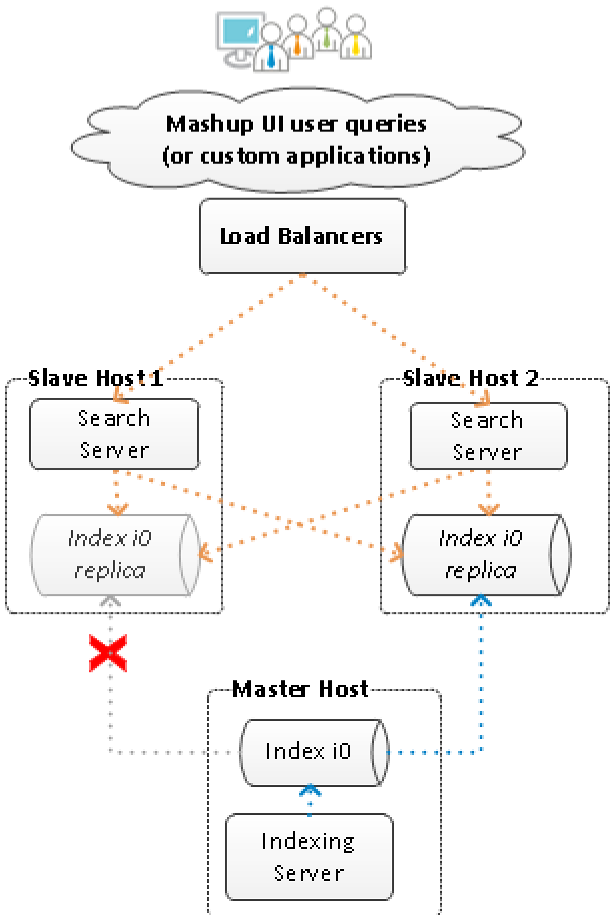


When index generation is complete on the primary server side, secondary servers receive the information that a new index generation is available.

secondary servers download new index files using a proprietary binary file transfer protocol.

Stop Index Replication

For maintenance or operation purposes, you may need to stop the replication process that is, detach replicas from the primary server.



1. Index replica on secondary server 1 has been detached from the primary server:
    - Modifications made to the primary server index are not replicated anymore.
    - Index replica on secondary server 1 is still available to process search queries.
2. Once modifications have been performed and the index has been reattached, the latest index generation available on the primary server are replicated on secondary server 1.

Detach Indexes from the Replication

1. If you need to make changes to the configuration (for example, search logic), detach the secondary server from the primary server:

```
./cvinit.[sh|bat] detach
```

2. Go to the API Console: http://<HOSTNAME>:<BASEPORT+1>/api-ui .
3. Click **Manage** to display the Management services.
4. To detach the index slice from the replication:
    - **Select MAMI indexing** .
    - Under **Operation**, select the `detachSliceReplica` method.

```
<DetachSliceReplica xmlns="exa:com.exalead.mercury.mami.indexing.v10"
              BuildGroup="bg0" indexSlice="0" sliceInstance="i1" />
```

5. Click **Save**.

The replication pauses but the indexing process continues. Alternatively, you can detach the index slice by clicking **Detach** in **Home > Indexing > Replication**.

Note: You can then reattach the index slice using the `attachSliceReplica` method. If you also detached the secondary server from the primary server, reattach the secondary server using `./cvinit.[sh|bat] attach`.

Monitoring Your License

Exalead CloudView licensing relies on a document token count. The total number of available tokens is determined when you purchase the product.

A token allows you to process a certain quantity of documents. This quantity depends on document types that are defined below. For example, 1 token allows the indexing of 1000 logs.

License monitoring features described below do not apply to Exalead CloudView deployments within the 3D#EXPERIENCE platform. Use the monitoring provided by the DS License Server instead.

Understand Document Types and Tokens

View Your License Details

Configure Email Alerts When Running Low on Tokens

Get New License Tokens or Add New Features

View User Tokens

Troubleshoot Issues with Licenses

Understand Document Types and Tokens

Exalead CloudView relies on a token-based license, which grants customers the right to process a limited number of documents, as indicated in the Product Portfolio. The following types of documents are available: Log, External Document, Internal Document, and Business Item.

The document type pushed by each connector is defined in the Administration Console. For information about document type settings, see "Setting up the Document Type Pushed by a Connector" in the Exalead CloudView Connectors Guide.

Document type	Description	Document cost per token
Log	A structured record logging a single event in a system with no rich attribute.	1 token = 1000 logs

Document type	Description	Document cost per token
	Examples: • an HTTP server log line, • a single SAP transaction, • a single line of a purchase receipt, • an email (excluding attachments).	
External Document	An unstructured document crawled from websites, which do not belong to the customer. Examples: • an HTML web page, • a PDF document crawled from a website that does not belong to the customer.	8 tokens = 1000 external documents
Internal Document	An unstructured document created by the customer. Examples: • an internal wiki page, • a page of an internal knowledge base portal (even if retrieved using the web crawler), • bulk office documents and files on employee hard disk drives.	14 tokens = 1000 internal documents
Business Item	A business document, work product or commodity, used in business operations by the customer. Examples: • an article in ERP with quantities, price, sourcing; • a customer record in CRM with inventory, contracts; • one ERP item in a CSV file grouping a set of ERP items;	60 tokens = 1000 business items

Document type	Description	Document cost per token
	• a structured office document, for example, a reference document or final version of a document; • a manufactured product processed or sold by the company: part, assembly with requirements, material, weight etc.	

Note: Consolidated documents are not taken into account. Exalead CloudView only counts the tokens of the raw documents entering the Consolidation Server, not those of the documents coming out of it.

View Your License Details

Follow the procedure below to view your license details.

1. In the Administration Console, select **Help > License**.
2. In the **Summary** section, verify your license **Expiration** date.
3. In the **Tokens** section, verify the number of used document tokens.

   The overall number of **Used tokens** is displayed as: **<number of used tokens>/<total number of license tokens available>**. The graph only shows the document tokens used by the Exalead CloudView instance.
4. In the **Document types** section, verify the document types. The cost of indexed documents depends on the document type.
5. In the **Features** section, verify the available licenses for your features and add-ons.
6. In the **Connectors** section, verify the list of connector licenses installed.
7. In the **Search users** section, verify the list of search users allowed:
   - **find**: users allowed to perform search on pages tagged as 'find' in the Mashup Builder.
   - **decide**: users allowed to perform search on pages tagged as 'decide' in the Mashup Builder.

Configure Email Alerts When Running Low on Tokens

By default, the Administration Console displays warnings on the UI and in the logs when you reach the maximum number of used tokens.

- A first message when you reach 80% of tokens.

- Another message when you reach 100% of tokens: `max tokens limit reached, indexing will be soon disabled`

  Important: You have a grace period of 15 days to index documents provided you do not exceed 20% more tokens.

You can also configure an email alert to get a notification when reaching a certain number of tokens in use, as described in the following procedure.

1. In the Administration Console, go to **Search > Reporting**.
2. In **Notifications**, select **Enable**.
3. Select **For license**.
4. In **Token alerting threshold**, specify the percentage of used license tokens that triggers an alert.
5. Click **Apply**.

Get New License Tokens or Add New Features

To get new license tokens or add new features, you must update your license.

1. Ask your account manager for a new license.
2. In the Administration Console, go to **Help > License**.
3. Click **Upload new license**.
4. Click **Browse** and select your new `.dat` file.
5. Click **Upload**.

View User Tokens

You can retrieve the number of users for the Mashup application.

1. Go to the `<DATADIR>\bin` directory and run:

   ```
   cvadmin licensing dump-users
   ```

   It returns the values for: `User id`, `User type`, and `Last access timestamp`.

Troubleshoot Issues with Licenses

Wrong Host ID

Problem: "I just installed Exalead CloudView and I get the message **CloudView is running on an unauthorized host. Indexing will soon be disabled."**

Cause: You installed Exalead CloudView on a host that is different from the host ID specified in your license file. Host IDs are linked to your hardware, and especially motherboards.

Solution: Check your server host ID and ask support for a new license file if your host ID has changed.

Expired License

Problem: "I get the message **Your license has been expired for 10 days. Indexing is now disabled**."

Cause: When your license expires, you are still able to work in Exalead CloudView during 14 days. Then you cannot access index documents anymore.

Solution: Connect to `http://www.3ds.com/terms/software-keys/` and ask for a new license. Follow the procedure in Get New License Tokens or Add New Features to upload it.

Too Many Users

Problem: "I see a warning in the log: " Too many users for type find" (or "Too many users for type decide"). What can I do?"

Solution: It means that you reached the limit of "find" or "decide" users as defined in your license key. User types are selected in the page properties for a given page in the Mashup Builder.

Backup/Restore Operations

You can back up your Exalead CloudView data by copying the `<DATADIR>` from your product installation. If you applied patches to your product installation, you can also back up the `<INSTALLDIR>`, to avoid reinstalling them. Procedures are available for both the API Console and the `cvcommand` command-line tool.

Backup Procedure

Restore Data

High Availability Backup Procedure

High Availability Restore Procedure

Backup Procedure

Standard Backup Operation

1. Stop Exalead CloudView on all hosts.

2. On all hosts, copy the `<DATADIR>` from your product installation. If you applied patches, you can also copy the `<INSTALLDIR>`.

   To reduce disk copy time, do not copy the `<DATADIR>/run` folder (containing logs).

Hot Backup Operation

1. To perform hot backup operations, you can use either the API Console or the `cvcommand` tool.

Hot Back Up Data Using the API Console

This is the procedure to follow if you use the Consolidation Server in your configuration.

1. In the API Console, select **Manage**.
2. If your Exalead CloudView configuration uses a crawler, stop it using:
   a. **Select MAMI crawl**.
   b. Under **Operation**, select the `stopCrawl` method.
   c. Click **Send**.
3. Freeze the product services:
   a. **Select MAMI master**.
   b. Under **Operation**, select the `freeze` method.
   c. Click **Send**.

      Note: While in freeze mode, all push operations are suspended, no indexing jobs are created and trigger job conditions are ignored.
4. Verify that Consolidation processes are disabled using:
   a. Under **Operation**, select the `getStatus` method.
   b. Specify the `instanceName` parameter (see previous step).
   c. Click **Send**.
   d. Specify the following to `enabled="false"`: `PushAPIStatus`, `TransformationStatus`, `AggregationStatus`.
5. Verify that processes are disabled:
   a. **Select MAMI indexing**.
   b. Under **Operation**, select the `getBuildGroupStatus` method.
   c. Specify the `indexTimeout` parameter to `0`.
   d. Click **Send**.
   e. Specify the following to `enabled="false"`: `PushServerStatus`, `AnalysisStatus`, `ImportStatus`.

6. On all hosts, copy the `<DATADIR>` from your product installation. You can also copy the `<INSTALLDIR>` if you applied patches.

   To reduce disk copy time, do not copy the `<DATADIR>/run` folder (containing logs).

7. Unfreeze the product services:
   a. **Select MAMI master**.
   b. Under **Operation**, select the `unfreeze` method.
   c. Click **Send**.

8. You can now restart your crawler if your Exalead CloudView configuration uses one.
   a. **Select MAMI crawl** .
   b. Under **Operation**, select the `startCrawl` method.
   c. Click **Send**.

Hot Back Up Data Using the `cvcommand` Tool

1. Go to the `<DATADIR>\bin` directory.
2. If your Exalead CloudView configuration uses a crawler, stop it using:
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/crawl stopCrawl crawlerName=mycrawler
   ```
3. Freeze all services using the master **freeze** method:
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/master freeze
   ```
   Note: The freeze mode suspends all push operations, and the creation of indexing jobs. It also ignores trigger job conditions.

4. Verify that processes are disabled using:
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/indexing getBuildGroupStatus
    buildGroup=<BUILDGROUP>
   ```
5. Check that the following are set to `enabled="false"`: `PushServerStatus`, `AnalysisStatus`, `ImportStatus`.
6. On all hosts, copy the `<DATADIR>` from your product installation. You can also copy the `<INSTALLDIR>` if patches were applied.

   Note: To reduce disk copy time, do not copy the `<DATADIR>\run` folder (containing logs).

7. Unfreeze the services using the master **unfreeze** method. For example:
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/master unfreeze
   ```
8. You can now restart your crawler if your Exalead CloudView configuration uses one.
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/crawl startCrawl crawlerName=mycrawler
   ```

Restore Data

When required, you can stop the product and then restore data.

Note: For the restore, use a data directory that was generated on the same platform architecture.

1. Stop Exalead CloudView on all hosts (main and secondary servers).
2. Restore data on all hosts:
   - Delete the content of the existing `<DATADIR>`.
   - Copy/paste the backup data within the `<DATADIR>`.
3. If you backed up the `<INSTALLDIR>`, restore it on all hosts.
4. Start Exalead CloudView on all hosts (main and secondary servers).
5. Check processes status using:

   ```
   <DATADIR>\bin\cvinit.[bat|sh] status
   ```

High Availability Backup Procedure

You can use this procedure to reduce backup time ONLY for Exalead CloudView HA deployments where the slave hosts are simple duplicates of your master host.

Warning: Do not use this procedure if the slave hosts support the **Storage Service** role.

Back Up Data on the Master Host

1. Go to the `<DATADIR>\bin` directory.
2. If your Exalead CloudView configuration uses a crawler, stop it using:

   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/crawl stopCrawl crawlerName=mycrawler
   ```
3. Freeze all build groups with the `cvcommand` tool:
   a. Go to the `<DATADIR>\bin` directory.
   b. Freeze the services using the master **freeze** method. For example:

      ```
      cvcommand <HOSTNAME>:<BASEPORT+11> /mami/master freeze
      ```

      Note: While in freeze mode, all push operations are suspended, no indexing jobs are created and trigger job conditions are ignored.
4. Disable that processes using:

   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/indexing getBuildGroupStatus
                buildGroup=<BUILDGROUP>
   ```
5. Define the following to `enabled="false"`: `PushServerStatus`, `AnalysisStatus`, `ImportStatus`.

6. Back up the `<DATADIR>` and the `<INSTALLDIR>`. To reduce disk copy time, do not copy the `DATADIR>/run` folder (containing logs).
7. Unfreeze all build groups. For example:
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/master unfreeze
   ```
8. You can now restart your crawler if your Exalead CloudView configuration uses one.
   ```
   cvcommand <HOSTNAME>:<BASEPORT+11> /mami/crawl startCrawl crawlerName=mycrawler
   ```

Back Up Data on the Slave Hosts

On the slave hosts hosting the search servers:

1. Back up the `<INSTALLDIR>` if you applied patches. Saving the `<INSTALLDIR>` does not require freeze/unfreeze steps.

High Availability Restore Procedure

You can use this procedure to reduce backup time ONLY for Exalead CloudView HA deployments where the secondary servers are simple duplicates of your primary server.

Warning: Do not use this procedure if the secondary servers support the **Storage Service** role.

Restore Data on the Primary Server

1. Stop Exalead CloudView.
2. Restore data:
   a. Delete the content of the existing `<DATADIR>`.
   b. Copy/paste the backup data within the `<DATADIR>`.
3. If you backed up the `<INSTALLDIR>`, restore it.
4. Start Exalead CloudView.
5. Check processes status using:
   ```
   <DATADIR>\bin\cvinit.[bat|sh] status
   ```

Restore Data on the Secondary Servers

On the secondary servers hosting the search servers and index replicas:

1. Stop Exalead CloudView.
2. Delete the existing `<DATADIR>`.
3. If you backed up the `<INSTALLDIR>`, restore it.
4. Reinstall the secondary server. See "Install the slave host".
5. Start Exalead CloudView.

Securing Exalead CloudView

This section describes how to secure your application, use certificates, and implement HTTPS.

General Recommendations

Follow these general recommendations to secure Exalead CloudView.

- Restrict access to the Exalead CloudView administration tools (all consoles accessible from `<BASEPORT>+1`) and services (accessible from `<BASEPORT>+11`) from external or public networks.
- Use a reverse proxy to hide hosts and service names from the outside (especially the Search API).
- Use SSL/TLS connections between applications and APIs.
- Use HTTPS and SSL to enhance Exalead CloudView security.

Securing your installation with HTTPS and SSL

This section describes how to manage certificates and enable HTTPS on your Exalead CloudView installation.

Managing Certificates

Exalead CloudView allows you to use secure HTTPS protocols (except for the Push API) that use strong encryption and digital authenticity certificates to protect data transmissions.

About Certificates

At installation time, the Exalead CloudView installer generates a self-signed certificate.

However, it is best to generate your own certificate signed by a trusted certificate authority for production located at:

- The public certificate, `DATADIR/security/<hostname>-<instance>.cert` for example, `DATADIR/security/server001.exalead.com-cvdefault.cert`
- The Private key, `DATADIR/security/<hostname>-<instance>.key` for example, `DATADIR/security/server001.exalead.com-cvdefault.key`

Public Certificate

The public certificate is saved in a DER format to facilitate the integration with Java tools (certification import/export in trust stores, etc.).

The requirements for the certificate are as follows:

- Generated key length must be: 2048 bits.
- Modulus must be: 2048 bits.
- SHA-2 certificates are supported.

Private Key

The Private key (`.key` file) is stored in a standard not encrypted `PEM` file format. The generated key length must be 2048 bits. It must use the following headers and footers:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

Import an Existing Certificate

Exalead CloudView allows you to use your own certificate signed by a trusted certificate authority.

The following properties are required:

- For the public certificate located at `DATADIR/security/<hostname>-<instance>.cert`:
  - Generated key length must be: 2048 bits.
  - Modulus must be: 2048 bits.
  - Stored in a DER file format.
- For the private Private key located at `DATADIR/security/<hostname>-<instance>.key`:
  - The Private key (`.key` file) is stored in a standard not encrypted PEM file format. The generated key length must be 2048 bits. It must use the following headers and footers:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

1. Check the certificate format using the following command:

   `openssl x509 -in <infile.cert> -text -inform <format>` (where format is DER or PEM depending on your needs)

2. Private keys are usually stored in encrypted PEM files. Convert them to a nonencrypted file. You can use openssl on the command line:

   ```
   openssl pkcs8 -topk8 -in <key> -out <hostname>-<instance>.key -nocrypt
   ```

3. Verify that the certificate and the private key (`.key` file) are stored using UNIX LF end of line characters:

   ◦ On Windows, you can use the following tool: `http://www.thefreecountry.com/tofrodos/index.shtml`.

   ◦ On UNIX, you can use `dos2unix`.

4. Overwrite the key and certificate files generated at installation time in `DATADIR/security`. If you are using an alias, the Private key name must use the alias and not the default `<hostname>-<instance>`. Performed this step on each product instance. These files are located at:

   ◦ The public certificate: `DATADIR/security/<hostname>-<instance>.cert`

   ◦ The Private key: `DATADIR/security/<hostname>-<instance>.key`

5. Add the server certificate to the truststore of every product instance:

   ```
   keytool -import -file <.cert file (DER)> -alias <jetty>
               -keystore DATADIR/security/trusted.servers.ks -storepass <exalead>
   ```

Enable HTTPS

You can enable HTTPS for internal connections, Exalead CloudView interfaces, Exalead CloudView APIs, and the Mashup UI. You can use the default certificates or define new ones.

Recommendation: When using HTTPS, to secure the Push API, set the authentication mode to **Basic** for the default connector in **Connectors > default > Deployment > Authentication > Mode**.

In the default Exalead CloudView configuration, only TLS is enabled for HTTPS connections. If you need SSL, edit restrictions on supported protocols in `<DATADIR>/config/DeploymentInternal.xml`.

Enable HTTPS for Internal Connections

Enable HTTPS for Exalead CloudView Interfaces

Exalead CloudView APIs

Enable HTTPs for the Mashup UI

Enable HTTPS for Internal Connections

You can use a proxy or network rules to secure internal ports. Alternatively, you may change the default product security as specified below.

1. Edit the `ProductSecurity.xml` file in `<DATADIR>/config`.
2. Specify the `secureInternalConnections` parameter to `true`.

Enable HTTPS for Exalead CloudView Interfaces

This section describes how to enable HTTPS for Exalead CloudView interfaces.

Import your certificate in the keystore, as described in Import an Existing Certificate, otherwise the Exalead CloudView interfaces cannot communicate with one another.

1. In the Administration Console, create a security source in **Search > Security Sources**. For details on configuring a security source, see Configuring Security Sources.
2. Go to the API Console URL: http://<HOSTNAME>:<BASEPORT+1>/api-ui.
3. Click **Manage**.
4. Select **MAMI adminui**.
5. Under **Configuration**, select the `setAdminUIConfig` method.
6. In `AdminUIConfig`:
   a. Define the `identityProvider` name.
   b. Define `useHttps` to `true`.
   c. Add `serverCertificate="your_certificate"` to specify your `.key` file.

   Example:

```
<AdminUIConfig serverCertificate="ngdev018.paris.exalead.com-cvdefault"
            useHttps="true" identityProvider="ip0"
            version="1405072169000">...</AdminUIConfig>
```

7. Click **Save**.
8. Select **MAMI master**.
9. Under **Configuration**, select the `setProductSecurity` method.
10. In `IdentityProviderConfig`, add `securitySource="MySecuritySource"` to specify the security source you defined at step 1 .

    Example:

```
<IdentityProviderConfig sessionInactivityTimeoutS="21600" securitySource="test"
            name="ip0"/>...
```

11. Click **Save**.
12. Click **Apply**.
13. Restart Exalead CloudView.

Activating HTTPS prevents the use of the `cvcmd` tool. Edit your `<DATADIR>/bin/ngstart.env` file, and replace `NGGATEWAYPROTOCOL=http` by `NGGATEWAYPROTOCOL=https`.

Exalead CloudView APIs

The following procedures describe how to enable HTTPS on the Exalead CloudView APIs.

Import your certificate in the keystore, as described in Import an Existing Certificate, otherwise the Exalead CloudView interfaces cannot communicate with one another. For all passwords specified in the procedures below, you can use encrypted passwords. For more information, see "Encrypt passwords" in the Exalead CloudView Installation Guide.

Enable HTTPS for the Search API

1. Go to the API Console URL: http://<HOSTNAME>:<BASEPORT+1>/api-ui.
2. Click **Manage**.
3. Select **MAMI master**.
4. Under **Configuration**, select the `setProductSecurity` method.
5. Add the following above `<master:IdentityProviderConfig>`:

   ```
   <master:SearchAPISecurity serverCertificate="my_certificate" useHttps="true"
   password="my_password" login="my_login"/> ...
   ```

   - `my_certificate` is the name of the `.key` file deployed in `<DATADIR>/security`.
   - `useHttps` value is `true`.
6. Click **Save**.
7. Click **Apply**.
8. Restart Exalead CloudView.

Activating HTTPS prevents the use of the `cvcmd` tool. Edit your `<DATADIR>/bin/ngstart.env` file, and replace `NGGATEWAYPROTOCOL=http` by `NGGATEWAYPROTOCOL=https`.

Enable HTTPS for the Management API

1. Go to the API Console URL: http://<HOSTNAME>:<BASEPORT+1>/api-ui
2. Click **Manage**.
3. Select **MAMI master**.
4. Under **Configuration**, select the `setProductSecurity` method.
5. Add the following above `<IdentityProviderConfig>`:

```
<master:MAMISecurity serverCertificate="my_certificate" useHttps="true"
password="my_password" login="my_login"/>...
```

- `my_certificate` is the name of the `.key` file deployed in `<DATADIR>/security`.
- `useHttps` value is `true`.

6. Click **Save**.
7. Click **Apply**.
8. Restart Exalead CloudView.

Activating HTTPS prevents the use of the `cvcmd` tool. Edit your `<DATADIR>/bin/ngstart.env` file, and replace `NGGATEWAYPROTOCOL=http` by `NGGATEWAYPROTOCOL=https`.

Enable HTTPS for the Push API

1. Go to the API Console URL: http://<HOSTNAME>:<BASEPORT+1>/api-ui
2. Click **Manage**.
3. Select **MAMI master**.
4. Under **Configuration**, select the `setProductSecurity` method.
5. Add the following above `<IdentityProviderConfig>`:

```
<master:PushAPISecurity serverCertificate="my_certificate" useHttps="true"
password="my_password" login="my_login"/>...
```

- `my_certificate` is the name of the `.key` file deployed in `<DATADIR>/security`.
- `useHttps` value is `true`.

6. Click **Save**.
7. Click **Apply**.
8. Restart Exalead CloudView.

Activating HTTPS prevents the use of the `cvcmd` tool. Edit your `<DATADIR>/bin/ngstart.env` file, and replace `NGGATEWAYPROTOCOL=http` by `NGGATEWAYPROTOCOL=https`.

Enable HTTPs for the Mashup UI

This section describes how to enable HTTPS in your Mashup UI application.

Import your certificate in the keystore, as described in Import an Existing Certificate, otherwise the Exalead CloudView interfaces cannot communicate with one another.

1. In the Administration Console, select **Deployment > Roles > Search Server** role.
2. Select **HTTPS**.
3. In **SSL certificate**, enter the name of the `.key` file deployed in the `<DATADIR>/security` folder.

The Mashup UIs will use this key. This overrides the certificate installed by default. For example, if your key file name is:

- `<hostname>-<instance>.key`, leave this field empty.
- `my_alias.key`, enter `my_alias` in this field.

Important: The `<DATADIR>/security` folder must also contain the `my_alias.key` and `my_alias.cert` certificate files. Make sure to import `my_alias.cert` in the trusted keystore with the `.cert` extension. If your certificate has a `.cer` extension, change it before importing it inside the keystore.

4. Restart Exalead CloudView.

Configure SSL Cipher Suites

When using HTTPS, you may need to set the cipher suites used by Exalead CloudView to include a specific cipher suite or exclude a cipher suite that is too weak to use.

By default, the Java Virtual Machine provides the cipher suites that Exalead CloudView uses. For more information about available cipher suites, see the JSSE Provider documentation.

The default configuration enables a few modern ciphers only. If you encounter issues with a specific browser or if you need to modify these restrictions to comply with your security policies, edit the list in `<DATADIR>/config/DeploymentInternal.xml`.

Note: These restrictions were added to the Exalead CloudView default configuration in R2016xR1 and are not added automatically when migrating from a previous version.

Include Cipher Suites

1. Edit `DeploymentInternal.xml` in `<DATADIR>/config`.
2. Add the following content below `<CloudviewDeploymentInternalConfig...>`:

```
<CloudviewDeploymentInternalConfig...>
<ServerCiphers>
<Include name="cipher_to_include"/>
<Include name="cipher_to_include"/>
</ServerCiphers>
```

3. Specify the ciphers to include in `<Include name="..."/>`.
4. Restart Exalead CloudView.

Exclude Cipher Suites

1. Edit `DeploymentInternal.xml` in `<DATADIR>/config`.
2. Add the following content below `<CloudviewDeploymentInternalConfig...>`:

```
<CloudviewDeploymentInternalConfig...>
<ServerCiphers>
  <Exclude name="cipher_to_exclude"/>
  <Exclude name="cipher_to_exclude"/>
</ServerCiphers>
```

3. Specify the ciphers to exclude in `<Exclude name="..."/>`.
4. Restart Exalead CloudView.

Protecting Your Application from Web Attacks

This section describes best practices and recommendations to protect your application from web attacks.

Secure Custom Developments

This section describes how to secure your custom developments.

- Follow best practices and security recommendations as described at `http://www.owasp.org` when creating custom widgets.
- Pay attention to:
  - Data sent to the back-end. Escape user-input to prevent XSS vulnerabilities.
  - Data stored via the storage API (used for collaboration widgets like rating or tagging) because it is the only way to inject data in indexed documents.

Enable Cross-Site Request Forgery Protection (CSRF)

Cross-Site Request Forgery (CSRF) is an exploit where the attacker impersonates a valid user session to gain information or perform actions on behalf of the user.

In Exalead CloudView, protection against CSRF is implemented via a token. You can associate this token with an expiration date if required. You can configure it in the Mashup Builder (**General > Application properties**).

Note: If you need to apply CSRF protection to custom widgets using POST forms, add `<render:crsf />` within the `<form>` tag.

1. Go to the Mashup Builder (http://<HOSTNAME>:<BASEPORT+1>/mashup-builder).
2. In **General > Application properties**, select **Enable CSRF protection**.

   The **CSRF token lifetime** field displays.
3. Enter the validity (in minutes) of the token. If the token must always apply, leave the field empty.
4. Click **Apply**.

Enable Phishing Protection

By default, when using all Exalead CloudView administration interfaces (Administration Console, API Console, Mashup Builder, Business Console), redirection URLs are not secured.

We cannot know how you access Exalead CloudView (for example, behind a proxy) and therefore, we cannot set a default host.

The following procedure explains how to make sure that redirection URLs match host/port from a list of trusted hosts.

1. Open the API Console (`<HOSTNAME>:<BASEPORT>+1/api-ui/`).
2. Click **Manage**.
3. Select **setProductSecurity** .
4. Edit the `<trustedHost>` node to declare all trusted hosts (replace `HOST` and `PORT`):

```
<master:trustedHost>
  <bee:StringValue value="HOST1:PORT1"/>
  <bee:StringValue value="HOST2:PORT2"/>
  <...>
</master:trustedHost>
```
5. Click **Apply**.
6. Restart Exalead CloudView.

Enable Clickjacking Protection

Clickjacking is a malicious technique of tricking Web users into clicking something different from what they think they are clicking, potentially revealing confidential information.

Attackers may exploit the XFS (Cross Frame Scripting) vulnerability on Exalead CloudView UIs to load an attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS), and overlay the phishing content on a malicious page.

Clickjacking may affect all Exalead CloudView UIs:

- The configuration and monitoring consoles: Administration Console, Mashup Builder, Business Console, API Console, and Monitoring Console. To tackle security failure on these consoles, see our General Recommendations.
- The Mashup UI applications created with the Mashup Builder. By default, we do not prevent iframe embedding as we need to be able to embed the Mashup UI within an iframe for page/widget previews to work correctly in the Mashup Builder and the Business Console. Once applications are no longer in development mode (previews are no longer useful), you can prevent iframe embedding on your Mashup UI applications as described in the following procedure.

1. Go to the `<DATADIR>/webapps/360-mashup-ui/WEB-INF/` directory.
2. Edit the `response-header-filter.xml` file and uncomment the following lines.
   ```
   <match-url url=".*">
     <header key="X-Frame-Options" value="SAMEORIGIN" />
   </match-url>
   ```
3. Repeat these actions for your other Mashup UI applications, if any.
4. Restart Exalead CloudView.

Control IP Address Binding

You can control IP address bindings in Exalead CloudView to restrict network access. An environment variable `NGBINDTARGET` in `ngstart.env` allows you to control the IP binding.

You can allow:

- **all**: bind on * allowing all hosts to communicate
- **hostname**: bind on the host's hostname specified in config - defaults to V4 binding
- **localhost**: bind on 127.0.0.1 allowing only localhost communication, which is incompatible with multihost deployment
- **ip6-localhost**: bind on ::1
- bind on a valid host specification (which must be a hostname, IPv4, or IPv6) which restricts communication locally or with this host. If the address is not bindable, it is considered as a product fatal error.

Deactivating Roles for Production

Some roles deployed by default in Exalead CloudView are not used in production environment. According to your needs, you can safely disable them to enhance your platform security.

1. Go to **Administration Console > Deployment > Roles > Administration > Admin UI**
2. If not used, select:
   - **API console disabled**
   - **Service console disabled**
   - **Inspection console disabled**
3. Go to **Administration Console > Deployment > Roles > Administration**
4. If not used, remove:
   - Mashup Builder
   - Business Console

Managing User Access

This section describes how to manage user access to Exalead CloudView interfaces.

Understanding User Management in Exalead CloudView

You can configure user access to the following Exalead CloudView interfaces:

- Administration Console
- Business Console (see **Misc > Configuration > General > Privileges**)
- The Mashup UI applications created with the Mashup Builder. For more information, see "Adding Security to Your Application" in the Exalead CloudView Mashup Builder User's Guide.

To set up your user management policy, you need to:

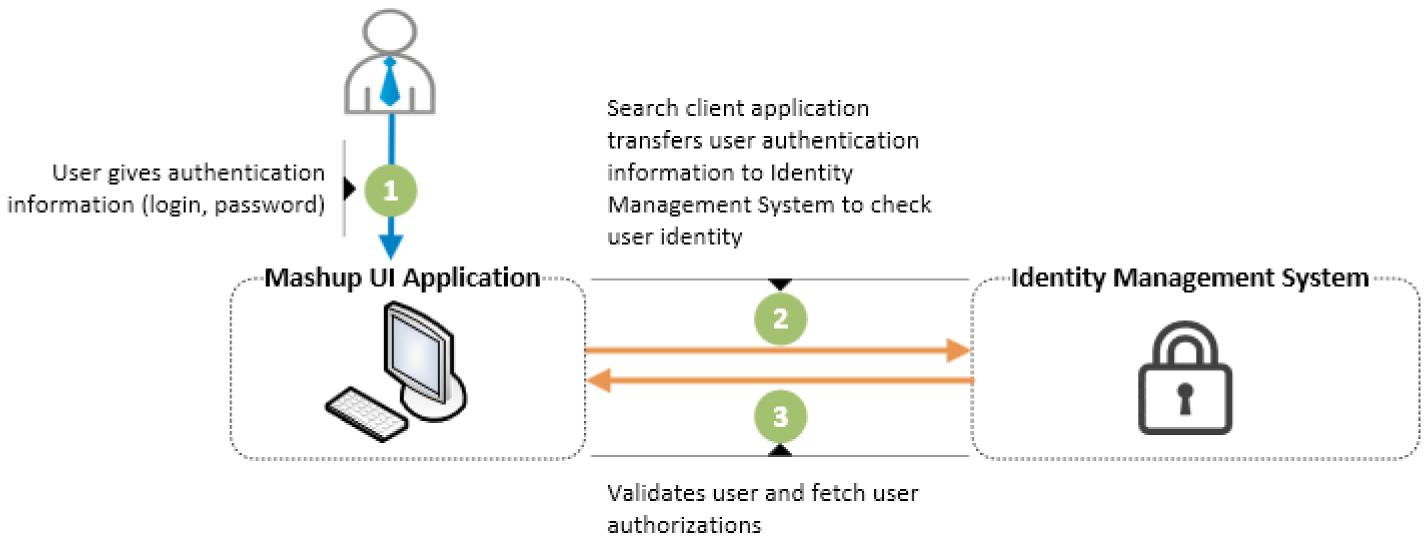
1. Configure the system used to authenticate users in the application using security sources. For example, you can create a simple security source or an LDAP security source. See Configuring Security Sources.
2. Define access rights to Exalead CloudView interfaces using roles. By default, there are seven predefined roles. Each role contains a list of read/write permissions to screens in the application. See "Configure roles" in the Exalead CloudView Installation Guide.
3. Associate roles to users using security tokens. See Configuring User Access.

How Search Users Access Documents

This section describes how users can access documents.

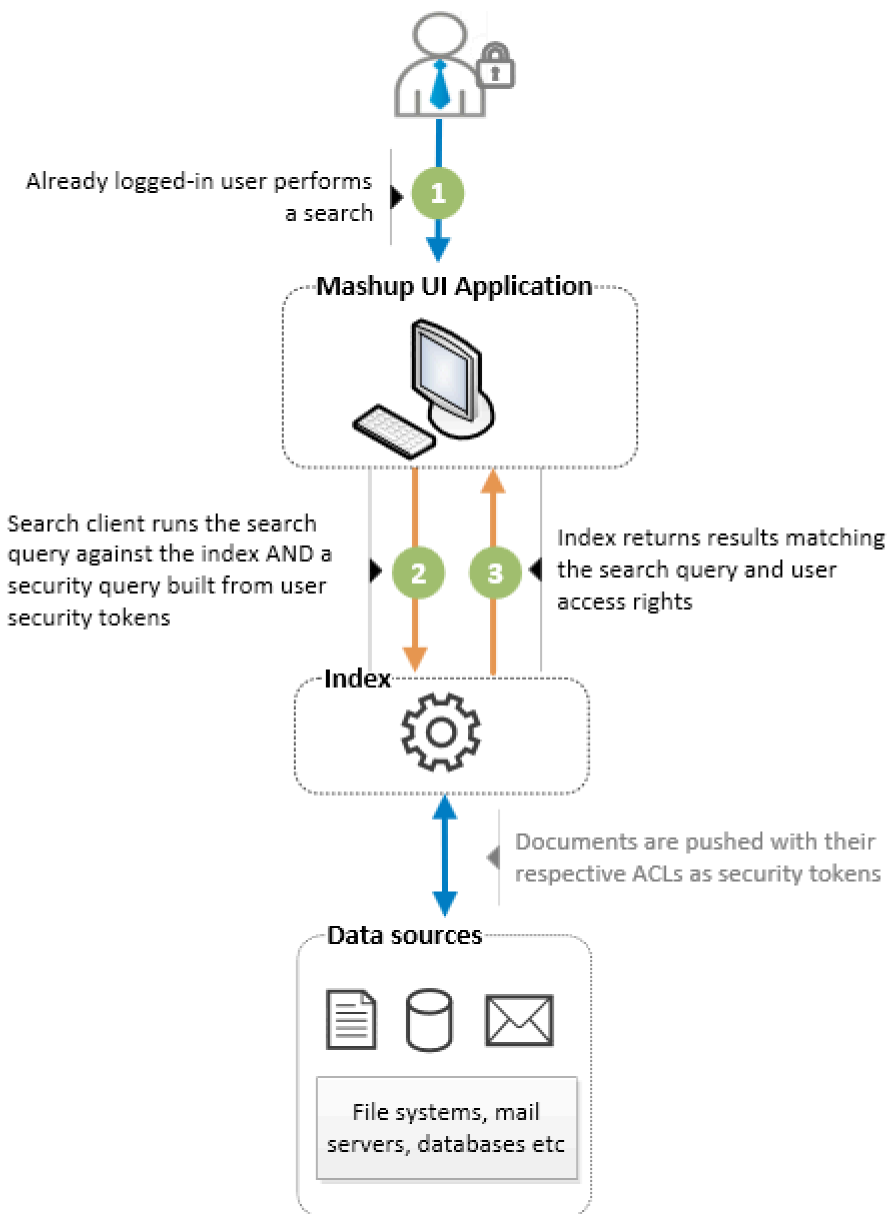
User Authentication

The schema below explains how the Mashup UI authenticates users.


User gives authentication information (login, password)
1
Search client application transfers user authentication information to Identity Management System to check user identity
Mashup UI Application
Identity Management System
2
3
Validates user and fetch user authorizations

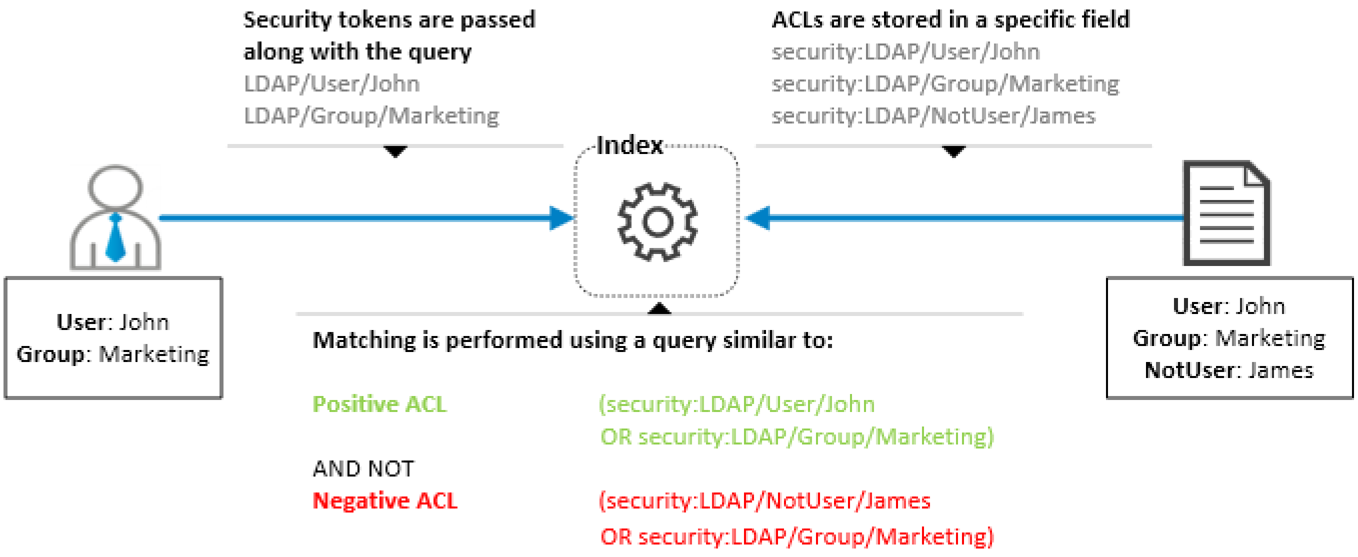
Search with Early Binding

The schema below shows the mechanism that guarantees that any search user accesses only the documents they are allowed to, because documents are indexed with their ACLs.


Already logged-in user performs a search
1
Mashup UI Application
Search client runs the search query against the index AND a security query built from user security tokens
2
3
Index returns results matching the search query and user access rights
Index
Documents are pushed with their respective ACLs as security tokens
Data sources
File systems, mail servers, databases etc

Document and Security Tokens

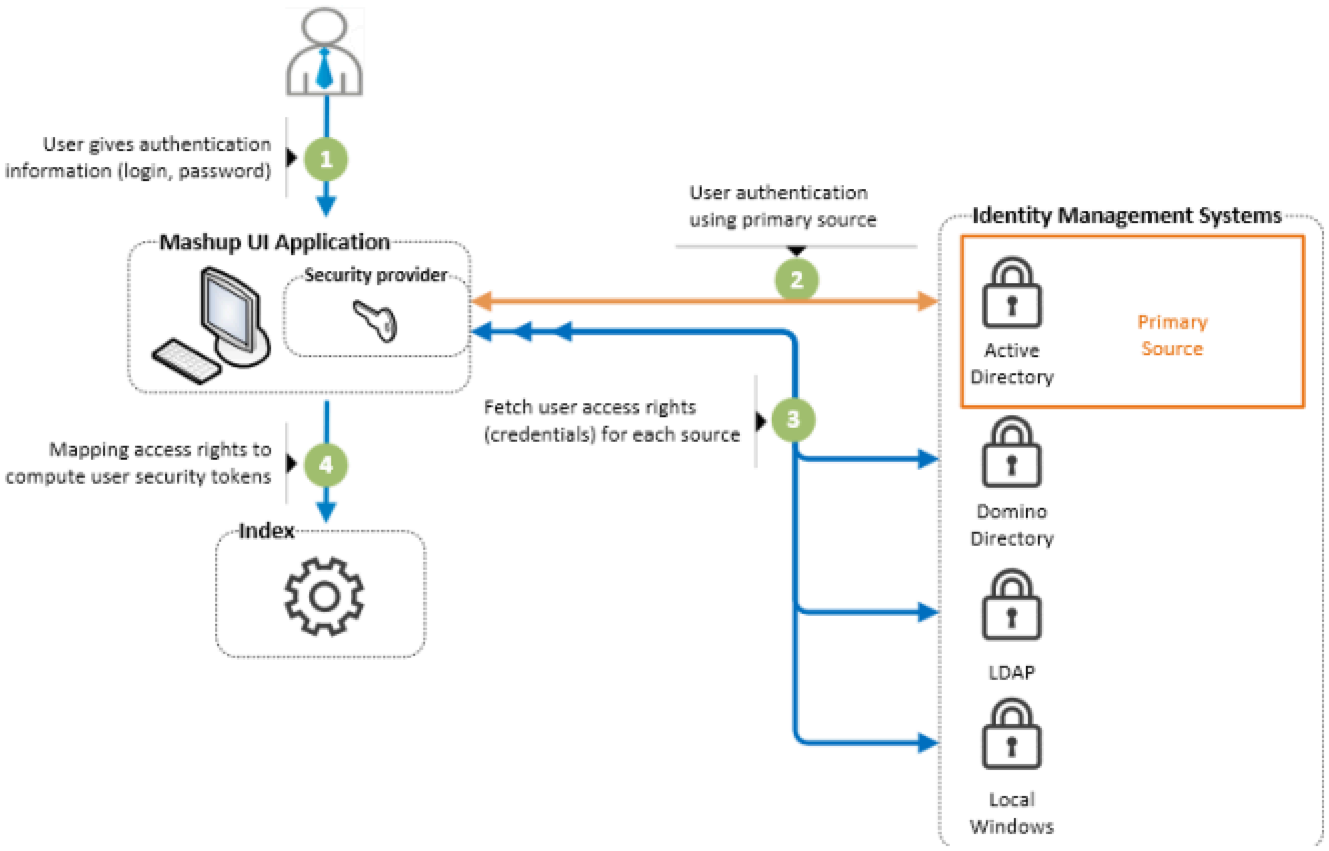
The schema below explains how user security tokens help to filter documents that the user cannot view/read. Exalead CloudView handles positive and negative ACLs.


Security tokens are passed along with the query
LDAP/User/John
LDAP/Group/Marketing
ACLs are stored in a specific field
security:LDAP/User/John
security:LDAP/Group/Marketing
security:LDAP/NotUser/James
Index
User: John
Group: Marketing
User: John
Group: Marketing
NotUser: James
Matching is performed using a query similar to:
Positive ACL
(security:LDAP/User/John
OR security:LDAP/Group/Marketing)
AND NOT
Negative ACL
(security:LDAP/NotUser/James
OR security:LDAP/Group/Marketing)

Using Several Security Sources

You can map several identification management systems to Exalead CloudView.

- User authentication uses a primary source.
- Access rights for each source are combined to allow secure search on multiple systems.


User gives authentication information (login, password)
1
Mashup UI Application
Security provider
User authentication using primary source
2
Identity Management Systems
Active Directory
Primary Source
Fetch user access rights (credentials) for each source
3
Domino Directory
LDAP
Local Windows
Mapping access rights to compute user security tokens
4
Index

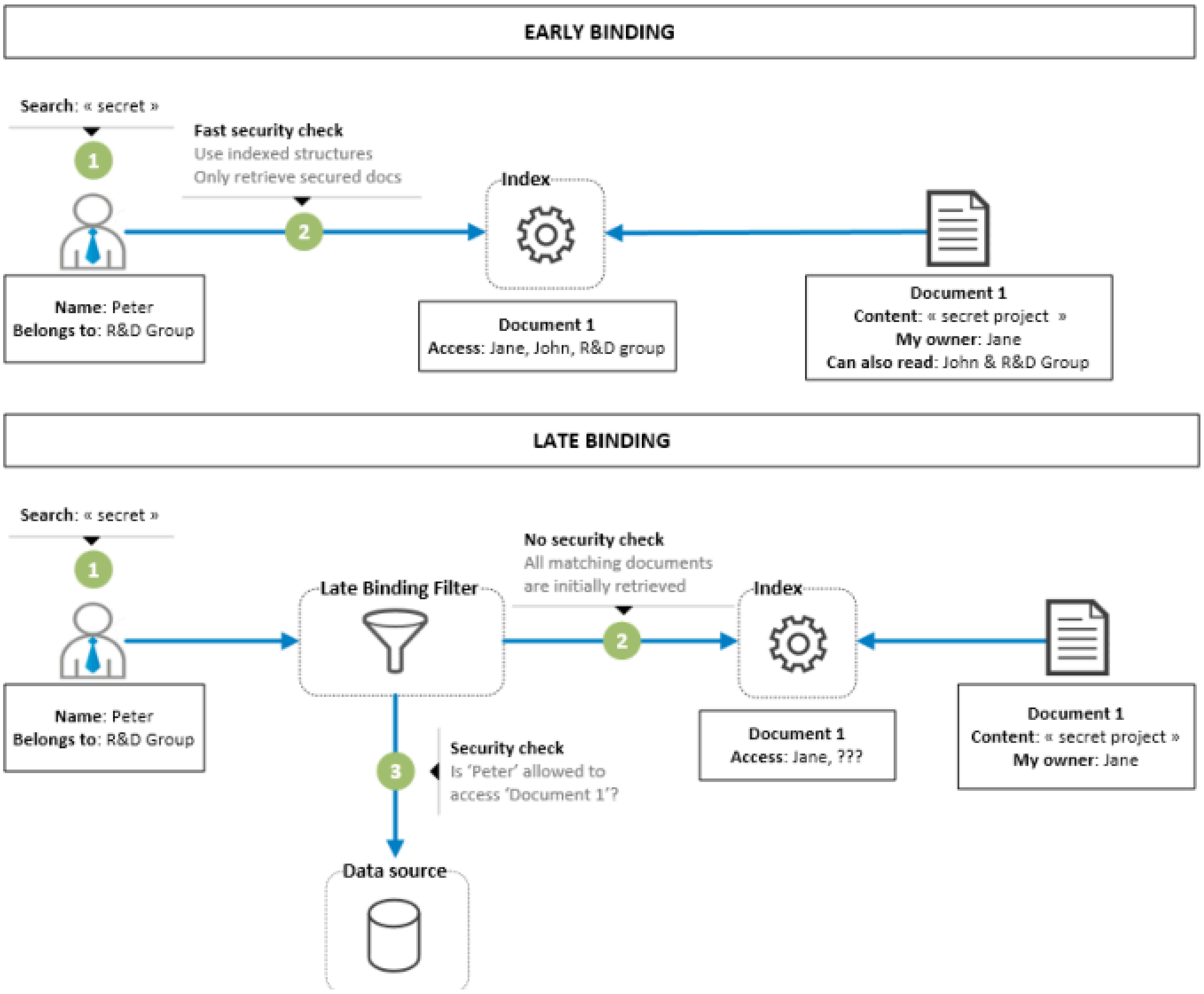
Late Binding Vs Early Binding

Security policies can be managed with more flexibility and early binding mechanism is not implemented.

As an alternative to early binding, you can use the late binding mechanism that checks document security tokens on the fly at search-time.

Note: Using late binding may significantly impact performance.

The schema below shows the differences between early and late binding mechanisms.



Configuring Security Sources

This section describes how to configure common security sources.

LDAP Security Source

LDAP Security Source

This section details properties and configuration procedures for the following LDAP-based security sources.

LDAP Overview

This section gives an overview of LDAP.

LDAP-Based Security Source

The LDAP-based security sources, as is the case for all Exalead CloudView security sources, provide two main functions:

- User authentication.
- User security tokens computing.

The former is possible only if user login functionality is enabled on the LDAP server. The latter authentication using an LDAP security source is a three-step procedure:

- User full DN retrieval: First, using the user login, which may be the full DN or any other valid login value, the security source tries to guess the user’s full DN.
- User identity validation: After the resolution of the user's full DN, the user's password is checked by opening an LDAP connection on the LDAP server. This action uses the user's full DN and the user's supplied password. If the connection is successfully opened, the computed user security tokens are returned.
- Security tokens computing: In the final step of authentication, the security source computes the security groups that the user belongs to and returns all the user security tokens to the application.

Security Servers

There are many implementations of LDAP. There are, however, several commonly deployed LDAP security servers. For these specific LDAP implementations, there are dedicated security sources in Exalead CloudView. The following is the list supported:

- OpenLDAP
- Active Directory (Microsoft windows user/group repository)
- Domino Directory (Lotus Domino user/group repository)
- Domino Directory (native)

There is also a generic LDAP security source that is fully configurable, allowing you to configure any LDAP server.

Authentication Method

The LDAP security source primarily supports the following authentication methods:

- None (anonymous access enabled)
- Simple

Other values are permitted depending on your configuration and the LDAP implementation. For more details, see `Context.SECURITY_AUTHENTICATION` on `http://docs.oracle.com/javase/jndi/tutorial/ldap/security/auth.html`.

Authentication Protocol

The LDAP security source supports the following authentication protocol versions:

- LDAP v2
- LDAP v3

Caching Group Information

The groups a user belongs to are using:

- Group parents attributes: the list of attributes that contain the parents of a given group.
- Group members attributes: the list of attributes that contain the members of a given group.
- Person parents attributes: the list of attributes that contain the parents of a given user.

Groups inclusion is the relationship between groups. When groups inclusion is configured in the LDAP server (`group-expand=true`), it is resolved once and set in the cache, as computation is very costly. It requires full LDAP groups listing. Use the Scheduler process to refresh the cache at an appropriate interval.

To refresh the source cache, click **Manage** and select MAMI master in the API Console. Under **Configuration**, select the **setSchedulingConfig** method and then edit the configuration:

```
<master:SetSchedulingConfig
xmlns:cdesc="exa:com.exalead.mercury.component.config.descriptor"
xmlns:secs="exa:com.exalead.security.sources.common"
xmlns:bee="exa:exa.bee"
xmlns:master="exa:com.exalead.mercury.mami.master.v10"
xmlns:config="exa:exa.bee.config">
<master:SchedulingConfig>
   <master:JobConfigGroup name="refresh_security_group">
      <master:DispatchJobConfig name="refresh_myldap">
         <bee:DispatchMessage messageName="ReloadSecuritySource"
         serviceName="/mami/master">
            <bee:messageContent>
               <bee:KeyValue value="myldap" key="securitySourceName"/>
            </bee:messageContent>
         </bee:DispatchMessage>
      </master:DispatchJobConfig>
   </master:JobConfigGroup>
   <master:TriggerConfigGroup name="refresh_security_group">
      <master:CronTriggerConfig name="refresh_myldap" startTime="0" endTime="0"
      jobGroupName="refresh_security_group"
      jobName="refresh_myldap" cronExpression="00 00 01 * * ?"/>
   </master:TriggerConfigGroup>
  </master:SchedulingConfig>
</master:SetSchedulingConfig>
```

Testing Authentication for a User

For all security sources, the Security Manager allows you to test the authentication for a user. When you configure a LDAP-based security source, you can test the user authentication on the configuration page with the **Test** button. This allows you to specify a Login and a Password for authentication. If the user is:

- Authenticated, the Display name and the security Tokens display.
- Not authenticated, an error message displays.

You can choose whether or not to perform a password check.

Note: Each call performs a new group cache computation and may take a few minutes, depending on the size of the LDAP source.

OpenLDAP Security Source

OpenLDAP is an open-source LDAP client and server implementation.

This section describes:

User Login

In the OpenLDAP, the `inetOrgPerson` object class identifies persons. User login match is case insensitive. A valid user login can use several values:

- Enable**DN Login**, for OpenLDAP, this allows the user to log in its full DN. Only DNs rooted on the defined LDAP search base are allowed.
- Attribute value `cn`, which is the common name.

There can be only one match for the value on the OpenLDAP server, otherwise, the login fails. The first step in the user login phase resolves the full user DN.

In some cases, the LDAP server login is not used and only security tokens are resolved.

Groups

In OpenLDAP, the `groupOfNames` object class identifies groups. To compute the groups a user belongs to, the `member` attribute defines the group entries.

Display Name

The `cn` attribute of the OpenLDAP builds the user display name.

Security Tokens Format

The security tokens are:

- `ldap:USERFULLDN`
- `ldap:USERGROUPFULLDN` for each group the user belongs to.

For example:

```
ldap:CN=Jane Smith,CN=Users,dc=office,dc=exalead,dc=com
ldap:CN=Sales,CN=Groups,dc=office,dc=exalead,dc=com
ldap:CN=Marketing,CN=Groups,dc=office,dc=exalead,dc=com
```

OpenLDAP Implementation

The OpenLDAP security source primarily supports the following authentication methods:

- None (anonymous access enabled)
- Simple

Other values are permitted depending on your configuration and the LDAP implementation. For more information, see `Context.SECURITY_AUTHENTICATION` on `http://docs.oracle.com/javase/jndi/tutorial/ldap/security/auth.html`.

OpenLDAP Software 2.x implements LDAP version 3 (default) and OpenLDAP Software 1.x implements LDAP version 2.

For more information, see .

Active Directory Security Source

The Active Directory is a Microsoft Windows-based implementation of LDAP. The Active Directory security source inherits the generic LDAP security source behavior with a specific configuration.

This section describes the Active Directory specific LDAP implementation.

Important: The Active Directory security source does not update security information automatically. The workaround to this limitation is to run the following command in your `<DATADIR>\bin` directory:

```
cvcommand <HOSTNAME>:<BASEPORT+11> /mami/master ReloadSecuritySource?
 securitySourceName="<SECURITY_SOURCE_NAME>"
```

User Login

In the Active Directory, the `user` object class identifies users. The user login match is case insensitive. You can use several attributes for a valid user login.

Attribute	Description
`DN login`	Allows the user to log in with the full DN. Only DNs rooted on the defined LDAP search base are allowed.
`sAMAccountName`	The windows account name, for example, `doe`.
`mail`	The user email address, for example, `john.doe@exalead.com`.
`userPrincipalName`	The user principal name, for example, `doe@office.exalead.com`.

Exalead CloudView tries to match the user login with these attributes in this order. There must be a unique match for the value on the Active Directory server, otherwise, the login fails. The first step in the user-login phase resolves the full user DN.

In some cases, the Active Directory server login is not used and only security tokens are resolved.

Display Name

The `displayName` attribute of the Active Directory generates the user display name.

Group Configuration

In Active Directory, the `group` object class identifies groups. The groups a user belongs to are computed at login time, using the configuration lists:

- **Group > Additional LDAP attributes**: the list of attributes that contain the additional attributes to return for a given group.
- **Group > Groups attributes**: the list of attributes that contain the members of a given group. To compute the groups a user belongs to, the `memberOf` attribute defines the group entries.
- **Person > Groups attributes**: the list of attributes that contain the groups of a given user.

Security Tokens

In Active Directory, user security tokens are generated based on the user entry and the group. Security tokens are built from windows object SIDs. The generated security tokens allow the matching of security tokens that can be set by a filesystem, Microsoft Exchange, or Microsoft Office Sharepoint Server connector.

Security tokens can be generated from the user entry. For example, if the user `CN=John Doe, OU=Exalead Users, DC=office, DC=exalead, DC=com` has the following LDAP attributes:

- `objectSid: S-1-5-19819-101018018-189989898`
- `primaryGroupId: 1891`

The following security tokens are generated:

- `windows:S-1-5-19819-101018018-189989898`
- `windows:S-1-5-19819-101018018-189989898-1891`
- `windows:S-1-5-32-545 (BUILTIN\Users group)`

Security tokens can be generated from groups to which the user belongs. For example, if the group `CN=Administrators, CN=Exalead Users, DC=office, DC=exalead, DC=com` has the following LDAP attributes:

```
objectSid: S-1-5-19819-101018018-189989898
```

The generated security tokens for this group is:

```
windows:S-1-5-19819-101018018-189989898
```

Lotus Domino Security Source

The Domino Directory is an LDAP implementation shipped within any standard installation of Lotus Domino. It is activated by default but you can deactivate it for custom setups.

The Domino Directory security source inherits the generic LDAP security source behavior with a specific configuration.

This section describes the Domino Directory specific LDAP implementation:

User Login

In the Domino Directory, the `dominoPerson` object class identifies persons. User login match is case insensitive. You can use several values for a valid user login:

Important: For Domino Directory, users can also log in using their full Lotus Notes DN. Write the DN in LDAP style, using comma separators. Specify `enableDNlogin` to `true`.

- Attribute `uid` is the user short identifier.
- Attribute `mail` is the user internet address.
- Attribute `cn` is the common name.

Exalead CloudView tries to match the user login with these attributes in this order. There must be a unique match for the value on the Domino Directory server, otherwise, the login fails. The first step in the user login phase resolves the full user DN.

Authentication Using the Domino Directory

If you want to authenticate the Exalead CloudView user with the Domino Directory security source, set the internet password on the Domino Directory. On the Domino Directory, activate the option to copy the user password from `notes.id` into the internet password of the user.

If the Domino Directory security source is used for authentication, the password entered by the Exalead CloudView user is compared to the internet password for the user profile in Lotus Domino.

Display Name

The `cn` attribute of the Domino Directory builds the user display name.

Groups

In Domino Directory, the `dominoGroup` object class identifies groups. To compute the groups a user belongs to, the `member` attribute defines the group entries.

Security Tokens

For a given user, the Domino Directory generates security tokens based on the user DN and the group.

Security tokens can be generated from the user DN. The user DN (LDAP format) is `CN=Jane Smith,CN=madvs001,DC=preprod`. The following security tokens are generated.

- `notes:builtin:-Default-`
- `notes:CN=Jane Smith/CN=madvs001/DC=preprod`

- `notes:*/CN=madvs001/O=preprod`
- `notes:*/O=preprod`
- `notes:*`

This automatic splitting on the user full DN is used for matching wildcard ACLs that could have been defined in the Lotus Notes databases that have been indexed. The separator `,` is replaced with `/` to match the security tokens that may have been set by the Lotus Notes connector.

Security tokens can be generated from groups to which the user belongs. The group DN (LDAP format) is `CN=Users,CN=madvs001,DC=preprod`. The following security tokens is generated.

- `notes:CN=Users/CN=madvs001/DC=preprod`

Limitations

Below is the list of Lotus Notes security features that this security source does not support:

- Handling of the security defined at the view level.
- Handling of the security defined at the item level.
- Handling of database ACL entries of type `Unspecified`. Only entries of type `Person` or `Group` are supported.
- Sequential checking of the ACL entries in the following order - user, group, and wildcard - until a match is found thus granting the highest access level and allowing access privileges at the same level to be grouped. Exalead CloudView Search denies access to a database if there is an ACL entry match at any level with no read access.

Configure an LDAP Security Source

You can follow the procedure below to configure an LDAP security source.

1. In the Administration Console, go to **Search > Security Sources** and click **Add security source**.
   a. In **Name**, enter a descriptive name.
   b. In **Type**, select an LDAP source type.
   c. Click **Accept**.
2. Go to the **Configuration** tab and verify that the **Deployed** check box is selected.
3. Under **Connection**, enter the LDAP-based Server **Address** for the security source. For example, `directory.exalead.com`.
4. Enter the login credentials in **Username** and **Password**.
5. Enter the LDAP's **Authentication method**.

   The default value corresponds to the default behavior for each security source.

6. Enter the LDAP **Protocol**.

   The default value corresponds to the default behavior for each security source.

7. Select an **Encryption** method.
8. Enter the LDAP's search **Base**. For example, `cn=Users,dc=office,dc=exalead,dc=com`.
9. Enter the **Timeout** value for the connection.
10. To make LDAP attribute matching case-insensitive, select **Ignore case**.
11. Click **Apply**.

Local Windows Security Source

This section describes the configuration of a local Windows security source on Windows platforms.

Technical Overview

The local Windows security source actually corresponds to the instance of a local security source in an Exalead CloudView installation on a Windows system (2000 / XP / 2003 / Vista / 2008).

The Windows local security source implementation relies on the Windows SSPI authentication tools used for managing user authentication on a Windows system. This means that every user with login rights to the system where Exalead CloudView is installed can be identified by this security source.

User Login

Users are identified by their Windows account name.

User Groups

The Windows group the user belongs to are returned.

Security Tokens

Security tokens are generated for each user. These security tokens may match the document's ACLs from a filesystem, Microsoft Exchange, or Microsoft Office Sharepoint Server connector.

There are security tokens generated for the user and group. The following are examples of security tokens generated from users and groups.

- If the user `smith` has an SID `S-1-5-19819-101018018-189989898`, the following security tokens are generated: `windows:S-1-5-19819-101018018-189989898`.
- If the group `Exalead` has an SID `S-1-5-19819-101018018-189989898`, the generated security tokens for this group are `windows:S-1-5-19819-101018018-189989898`.

Configuring the Security Source

The following parameters control the local windows connection used for implementing this security source.

- **Domain**: Domain used to identify the user.
- **login as a token**: To push the login as a security token.

```
<SecuritySource name="Windows"
classId="exa:com.exalead.security.sources.local.windows.WindowsLocalSecuritySource
  connectorServer="java0">
  <config>
      <KeyValue key="domain" value="exalead"/>
   </config>
</SecuritySource>
```

Local UNIX Security Source

The local UNIX security source actually corresponds to the instance of a local security source in a Exalead CloudView installation on a UNIX system.

This security source implementation relies on PAM subsystem's `password authentication module`. For more information, see your PAM configuration, in particular the `/etc/pam.d` configuration.

For a Local UNIX security source on a UNIX system, this section describes:

Technical Overview

User Login

Users are identified by their UNIX account name.

User Groups

The groups a user belongs to are computed from information found in the system user database.

Security Tokens

For a given user, the following security tokens are generated. These security tokens are built so that ACLs of documents indexed on a UNIX file system can be matched by Exalead CloudView on a UNIX system.

There are security tokens generated for the user and group. The following are examples of security tokens generated from users and groups.

- If the user UNIX account name is `smith` and the user identifier (UID) is `501`, the following security token are generated `unix:user:501`.
- If the group UNIX name is `exalead` and its UID is `601`, the following security token are generated `unix:group:601`.

Prerequisites

For this security source to be functional, the user running Exalead CloudView must have read access to the system user database.

Limitation

The UNIX security source is only compatible with NIS-based authentication.

Configuring the Security Source

The following parameters control the local UNIX connection used for implementing this security source.

- **Domain**: If empty, the default PAM `login` is used. Otherwise, the specified PAM service is used. The typical configuration is `/etc/pam.d/<service>`.
- **login as a token**: To push the login as a security token.

```
<SecuritySource name="Unix"
 classId="exa:com.exalead.security.sources.local.unix.UnixLocalSecuritySource"
 connectorServer="java0">
</SecuritySource>
```

Remote HTTP Security Source

This section explains how remote HTTP security works, as well as how to configure both the security on the source and on Exalead CloudView.

Exalead CloudView provides a remote security API based on a simple HTTP GET or POST / XML protocol. Remote security providers can therefore supply a way for Exalead CloudView to authenticate a user and to retrieve its security tokens. These security tokens are then used by Exalead CloudView to secure document access: the user can only see the documents that match user security tokens of this given user.

Mandatory Services

- `Authenticate (GET or POST)`: authenticates a user

  Must be available on `<URL>/authenticate` and on `<URL>/`.
- `Reset (GET)`: restores the security source to its initial state

Must be available on `<URL>/reset`.

Authentication Request

Exalead CloudView Search triggers a remote security query to the remote security connector using a simple HTTP GET or POST. Authentication parameters, `login` and `password`, pass through the url. If there is no need for authentication, the url parameter `checkPassword` with the value `false` can be passed and the password `parameter` is optional.

Authentication Response

Whether the remote security source succeeded to authenticate the user, it sends back a `200 OK` and embeds in the http body the authentication result (XML representation) with it is state.

If the authentication succeeded, it contains the list of security tokens of the user; otherwise it contains the cause of failure.

Required Response Format

The `AuthenticationResult` must be in XML, with the following attributes:

- `xmlns` (required): internally used by Exalead CloudView. The value is always `exa:com.exalead.security.sources.common`.
- `success` (required): it defines if the authentication succeeded. Possible values: `true` or `false`.
- `userId` (required): the login used to authenticate the user.
- `userDisplayName`: The real name of the authenticated user. Used only when authentication is successful.
- `cause`: the reason of failure. Used only when authentication fails.

Here is an example of the result format in XML.

```
<AuthenticationResult xmlns="exa:com.exalead.security.sources.common"
 userDisplayName="Anonymous user" userId="guest" success="true">
  <SecurityToken xmlns="exa:com.exalead.security.sources.common"
 token="remote:first_token" />
  <SecurityToken xmlns="exa:com.exalead.security.sources.common"
 token="remote:second_token" />
</AuthenticationResult>
```

Security Tokens

If authentication is successful, the node also contains a list of security tokens.

A `SecurityToken` has one attribute.

- `token`: the value of the security token, which must be the same value than the one used during indexing.

Configuring a Remote HTTP Security Source

In the Administration Console, go to **Search > Security Sources** and click **Add security source**.

- For **Type**, select **Remote Http**.
- The **isAlive path** parameter defines the path to a service that must reply with an http code below 400. When using multiple hosts, it helps Exalead CloudView to detect available hosts. Leave this field empty to disable this option.
- The **service path** parameter defines the path to access the security service.
- The **maxRetries** parameter defines the number of trials before giving up on a host connection.

To provide high availability, you can declare multiple hosts with the following parameters:

- The **Host** parameter defines the host name of the remote security provider.
- The **Port** parameter defines the port of the remote security provider.
- The **Priority** parameter is only used if you specify multiple hosts. Connections are made first on a host with higher priority. In case of equality, it is randomly selected.

Federate Several Security Sources

You can use security source dispatchers to federate several security sources.

Documents are coming from several different connectors. Each connector is associated to a security model with different credentials. The goal is to provide a unified page so that the user only needs to log on once, and see all the documents from all the sources the user can access to.

Define:

- The security source tokens to retrieve.
- For each security source:
  - Is the user who they claim to be?
  - What are the access rights linked to the user?

Security Source Logins

If the login is not identical on each source, you need to map the login entered by the user with the pattern that exists in all the other security sources. In this case, use the **Login rewriting** field and the `$login` variable.

Example:

John Doe is identified by `mydomain\jdoe` in the first security source and by `jdoe@mycompany.com` in the second security source.

To map both logins, enter in the **Login rewriting** field:

- `mydomain\$login` for the first security source
- `$login@mycompany.com` for the second security source

Authentication checks that the user exists and that their password is correct. It is mandatory for at least one security source.

Authorization retrieves access rights granted to the user (security tokens). Password is optional. If the password check fails, no access right is granted to the user for the security source.

Create a Security Source Dispatcher

1. In the Administration Console, go to **Search > Security Sources** and click **Add security source**.
2. In **Name**, enter a name for the security source.
3. In Type, select **Security source dispatcher**.
4. Click **Accept**.
5. In the **Configuration** tab, configure the **Authentication behavior**:
   - **First**: retrieves the tokens for the first security source on which the user successfully authenticates.
   - **Merge**: retrieves the tokens for **all** security sources on which the user successfully authenticates.
   - **No authentication**: removes the authentication constraint. This is useful when users connect through SSO.
6. In the **Configuration** tab, configure the **Additional tokens**: for example, if you need to create a group of users, add security tokens.
7. In the **Configuration** tab, configure the **Security sources**:
   a. **Name**: select the security sources you want to federate.
   b. **Login rewriting**: if required, set the login schema to use.
   c. **Source type**: select either the **authentication** or **authorization** mode.
   d. **Check password**: select this check box if you want to check the password when using the **authorization** security source.
   e. **Actions**: use the arrows to organize security sources.
8. Test user authentication to make sure that your security source dispatcher is properly configured.

Configuring Roles

This section describes how to configure roles.

What Is a Role

A role is a set of permissions made of:

- A `permission id`, corresponding to screens displayed in the Administration Console.
- A `permission`, corresponding to authorized actions (`write`, `read`, `none`) for a given `permission id`.

You can configure permissions in the `AdminUI.xml` file in `<DATADIR>\config\`. You can edit this file to:

Example of role and permissions configuration in `AdminUI.xml`:

```
<aui:AdminUIConfig identityProvider="ip0" useHttps="false"
 version="1501155968000"
 xmlns:aui="exa:com.exalead.mercury.mami.adminui.v10"
 xmlns:config="exa:exa.bee.config">
  <aui:Role name="corpus-manager">
   <aui:Permission id="home" permission="write"/>
   <aui:Permission id="about" permission="write"/>
   <aui:Permission id="collect-connectors" permission="write"/>
   <aui:Permission id="connector-list-widget" permission="write"/>
   <aui:Permission id="collect-connector" permission="write"/>
...
   </aui:Role>
...
</aui:AdminUIConfig>
```

It displays:

- A role: `<aui:Role name="corpus-manager">`
- Associated to a permission: `<aui:Permission id="collect-connectors" permission="write"/>`

It means that the users with the role have `corpus-manager` `write` permission to the **Collect > Connectors** page.

Default Roles

The following roles are available by default.

Role	Access to	Configured in
Administrator	**Administration Console**: Full access **Business Console**: Full access	Hard-coded
Corpus manager	**Administration Console** : Write access to connectors, build groups, data model, linguistics. **Business Console**: • No access to approval workflow (in **Semantic > Resources > Review & Publish tab > Approval** section) • No access to **Misc > Configuration** • No access to **Semantic > Resources**	`AdminUI.xml`
Deployment manager	**Administration Console** : • Write access to build groups. • Read access to connectors, security sources, linguistics. **Business Console**: • No access to approval workflow • No access to **Misc > Configuration** • No access to **Semantic > Resources**	`AdminUI.xml`
Search application manager	**Administration Console** : Write access to linguistic, security sources and search components. **Business Console**: • No access to approval workflow • No access to **Misc > Configuration** • No access to **Semantic > Resources**	`AdminUI.xml`

Role	Access to	Configured in
Linguist	**Administration Console**: No access **Business Console**: • No access to approval workflow • No access to **Misc > Configuration**	Hard-coded but can be declared in `AdminUI.xml` to add permissions
Business administrator	**Administration Console**: No access **Business Console**: No access to **Misc > Configuration**	Hard-coded but can be declared in `AdminUI.xml` to add permissions

Add Permissions to an Existing Role

The following procedures describe how to add permissions to an existing role.

Define Permission IDs

The following table lists available permissions IDs.

Permission ID	Access to
`about`	The **About** page
`access-search-api`	The **Search API** menu item
`access-search-logic`	The Search Logic's configuration page, for example, **Search Logics > sl0**
`access-search-logics`	The **Search Logics** menu item
`access-security-source`	The Security Source's configuration page
`access-security-sources`	The **Security Sources** menu item
`access-suggest`	The **Suggest** menu item
`build-groups`	The **Home** > **Indexing** operations
`buildgroups-build-group`	The **Deployment > Build Groups** tab
`buildgroups-search-targets`	The **Build Groups > Search Targets** tab
`connector-advanced-configuration-widget`	The Connector's **Advanced** configuration tab
`connector-config-widget`	The Connector's **Configuration** tab

Permission ID	Access to
`connector-deploy-widget`	The Connector's **Deployment** tab
`connector-logs-widget`	The HTTP Connector's **Logs** tab
`connector-operation-widget`	The Connector’s **Operation** tab
`consolidation`	The **Consolidation** menu item
`cvdiag`	The **Help > Create system report** page
`data-connector`	The Connector's configuration page, for example, **Connectors > Your_connector**
`data-connectors`	The **Connectors** menu item
`data-crawler`	The HTTP Connector's configuration page, for example, **Connectors > Your_HTTP_connector**
`data-model`	The **Data Model** menu item
`data-model-classes`	The **Data Model > Classes**
`data-model-document-processors`	The **Data Processing >** Pipeline name **> Document Processors** tab
`data-model-index-mappings`	The **Data Processing >** Pipeline name **> Mappings** and **Mapping Limits** tab
`data-model-processing`	The **Data Processing** tab
`data-model-schema`	The **Data Model > Advanced Schema** tab
`data-model-semantic-processors`	The **Data Processing >** Pipeline name **> Semantic Processors** tab
`data-model-semantic-types`	The **Data Model > Semantic Types** tab
`data-model-widget`	The **Data Model** tabs
`deploy-build-groups`	The **Build Groups** menu
`deploy-roles`	The **Roles** menu item
`diagnostic`	The **Diagnostics** page
`home`	The **Home** page
`index-tuning`	The **Tuning** page containing the analysis and compact policies, and commit trigger conditions configuration

Permission ID	Access to
`license`	The **License** page
`linguistic`	The **Linguistics** menu item
`linguistic-dictionary-edit`	The Dictionary config page, for example, **Dictionaries > dict0**
`linguistic-tokenization-edit`	The Tokenization config page, for example, **Tokenization > tok0**
`linguistic-tokenization-list`	The **Linguistics > Tokenization** list page
`logs`	The **Logs** page
`logs-level`	The **Default logging level** select box
`plugins`	The **Plugins** page
`processes`	The **Home > Processes** section
`linguistic`	The **Linguistics** page (list of tokenization configurations + dictionaries)
`linguistic-dictionary-edit`	The **Dictionary** page
`resources`	The **Resource** management page
`rollback`	The **Rollback** options displayed in the list on the right of **Apply**.
`search-logic-content-restriction`	A Search Logic's **Content Restriction** page
`search-logic-hit-content`	A Search Logic's **Hit Content** page
`search-logic-limits`	A Search Logic's **Limits** page
`search-logic-linguistics`	A Search Logic's **Query expansion** page
`search-logic-navigation`	A Search Logic's **Facets** page
`search-logic-query-language`	A Search Logic's **Query Language** page
`search-logic-relevance`	A Search Logic's **Sort & Relevance** page
`search-logic-virtual-fields`	A Search Logic's **Virtual Fields** page
`search-reporting`	The **Reporting** page

Permission ID	Access to
`security-source-advanced-widget`	The security source's **Advanced** page
`security-source-edit-widget`	The security source's **Configuration** page

Add a Permission

1. Edit the `AdminUI.xml` file in `<DATADIR>\config\`.
2. Add a permission to a role (see table above) using the following tag:

   `<Permission id="<permission_id>" permission="<read_write_none>"/>`

   For example, to give write access to the **Administration Console > Deployment > Resources > Plugins** section, use the following tag: `<Permission id="plugins" permission="write"/>`.

Create Custom Roles

You can create custom roles based on specific permissions. You can also use this procedure to add permissions to existing hard-coded roles.

Custom roles are automatically added to the list of available roles when configuring user access. For more information, see Assign Roles to a User.

1. Edit the `AdminUI.xml` file in `<DATADIR>\config\`.
2. Add your role below a `</aui:Role>` end-tag using the following syntax:

   `<aui:Role name="my_custom_role">`
3. Add permissions to the role using the following tags:

   `<aui:Permission id="<permission_id>" permission="<read_write_none>"/>`
   `</Role>`

   Example: To give write access to the **Administration Console > Deployment > Resources > Plugins** section to a new role called `plugin-admin`, use the following syntax:

```
<aui:Role name="plugin-admin">
  <aui:Permission id="plugins" permission="write"/>
</aui:Role>
```

Configuring User Access

Once roles are configured, you can create users and security tokens to manage user groups in the **Administration Console > [user logged in] > Users**. User configuration is stored in `<DATADIR>\adminui\users.xml`.

Users and security tokens are created dynamically. You do not need to save or apply changes.

Assign Roles to a User

You can follow the procedure below to assign roles to a user.

1. Go to **Administration Console > [user logged in] > Users.**
2. In the **Users** section, select a security source.
3. Click **Add user**.

   A dialog box appears.
4. Enter the user login as defined in your security source in **Administration Console > Search > Security source**:
   - For **Simple Security** sources, use the user login defined.
   - For other security sources, user login is displayed as **User id** when testing authentication in **Test user authentication**.
5. Select the roles to assign to the user.
6. Click **Accept**.

Assign Roles to a Group of Users

You can use security tokens to assign roles to a group of users.

1. Go to **Administration Console > [user logged in] > Users.**
2. In the **Users** section, select a security source.
3. Click **Add security token**.

   A dialog box appears.
4. Enter the security token as defined in your security source in **Administration Console > Search > Security source**:

- For **Simple Security** sources, security tokens are created with user login/password.
- For other security sources, security tokens are displayed as **Tokens** when testing authentication in **Test user authentication**.

5. Select the roles to assign to your security token.
6. Click **Accept**.

   The security token is applied to all users belonging to it.

Configure Access to Business Console Applications

The `config/360/<app>/BusinessConsoleConfiguration.xml` file grants the access to the Business Console Mashup UI applications.

You can restrict access rights per application in the Business Console under **Configuration > General > Privileges > Users that can access this application**. In that case, only the users mentioned in the field have access to the application.

The `config/360/undefined_appid/BusinessConsoleConfiguration.xml` file grants the access to the Search API application. It is not a Mashup UI application, but a mock application used to get statistics on the Search API directly. It works the same but by default, its access is restricted to the `admin` user.

```
<PrivilegesConfiguration>
  <KeyValue value="" key="canApplyConfiguration"/>
  <KeyValue value="admin" key="hasAccess"/> <!-- value="" means all users -->
</PrivilegesConfiguration>
```

The **Configuration > General > Privileges > Users that can access this application** option is not available for the Search API application. To add users, you must edit the `config/360/undefined_appid/BusinessConsoleConfiguration.xml` file and then apply the configuration manually.

Use Cases

This section describes detailed procedures for common use cases.

Give Full Access to an LDAP-Authenticated User

Give Access to Plugins to an Existing User

Give Full Access to an LDAP-Authenticated User

We need to give full access to the Administration Console to an LDAP-authenticated user.

Step 1: Create Connection to the LDAP Directory

1. Go to **Administration Console > Search > Security Sources.**
2. Click **Add security source**.
3. Enter the name of your LDAP security source and select the **Active directory** type.
4. In **Config > Connection**, enter LDAP connection information:
   - address (that is, `myoffice.mycompany.com`)
   - username
   - password
   - base (that is, `OU=people,DC=example,DC=com`)
5. Test your configuration in **Test user configuration**.
6. Keep the displayed **User id**. It serves to configure the user at step 2.

Step 2: Configuring User Access Rights

1. Go to **Administration Console > [user logged in] > Users.**
2. In the **Users** section, select a security source.
3. Click **Add user**.
4. Enter the user login as displayed previously when testing your security source.

   For example, `CN=John Doe,OU=people,DC=example,DC=com`.
5. Select the **Administrator** role and click **Accept**.

   To apply this access right to a group of users, add a security token corresponding to the tokens displayed when testing user configuration.

Give Access to Plugins to an Existing User

We need to give access to the **Plugins** section in the Administration Console to an existing user having a 'linguist' role (that is, no access to the Administration Console).

Step 1: Open the users.xml File

1. Open the `users.xml` file in `<DATADIR>\adminui\`.
2. Look for the user having the 'linguist' role.
3. Copy the tag corresponding to the role name: `<Role name="linguist"/>`.

Step 2: Edit the AdminUI.xml File

1. Open the `AdminUI.xml` file in `<DATADIR>\config\`.

2. Paste the tag corresponding to the role name after a `</aui:Role>` end-tag.
3. Add the following tags: `<aui:Permission id="plugins" permission="write"/> </aui:Role>`.
4. Restart Exalead CloudView.

The user now has access to the **Plugins** section in the Administration Console.

Monitoring Exalead CloudView

This section describes how to monitor the status of your Exalead CloudView installation.

About Exalead CloudView Monitoring

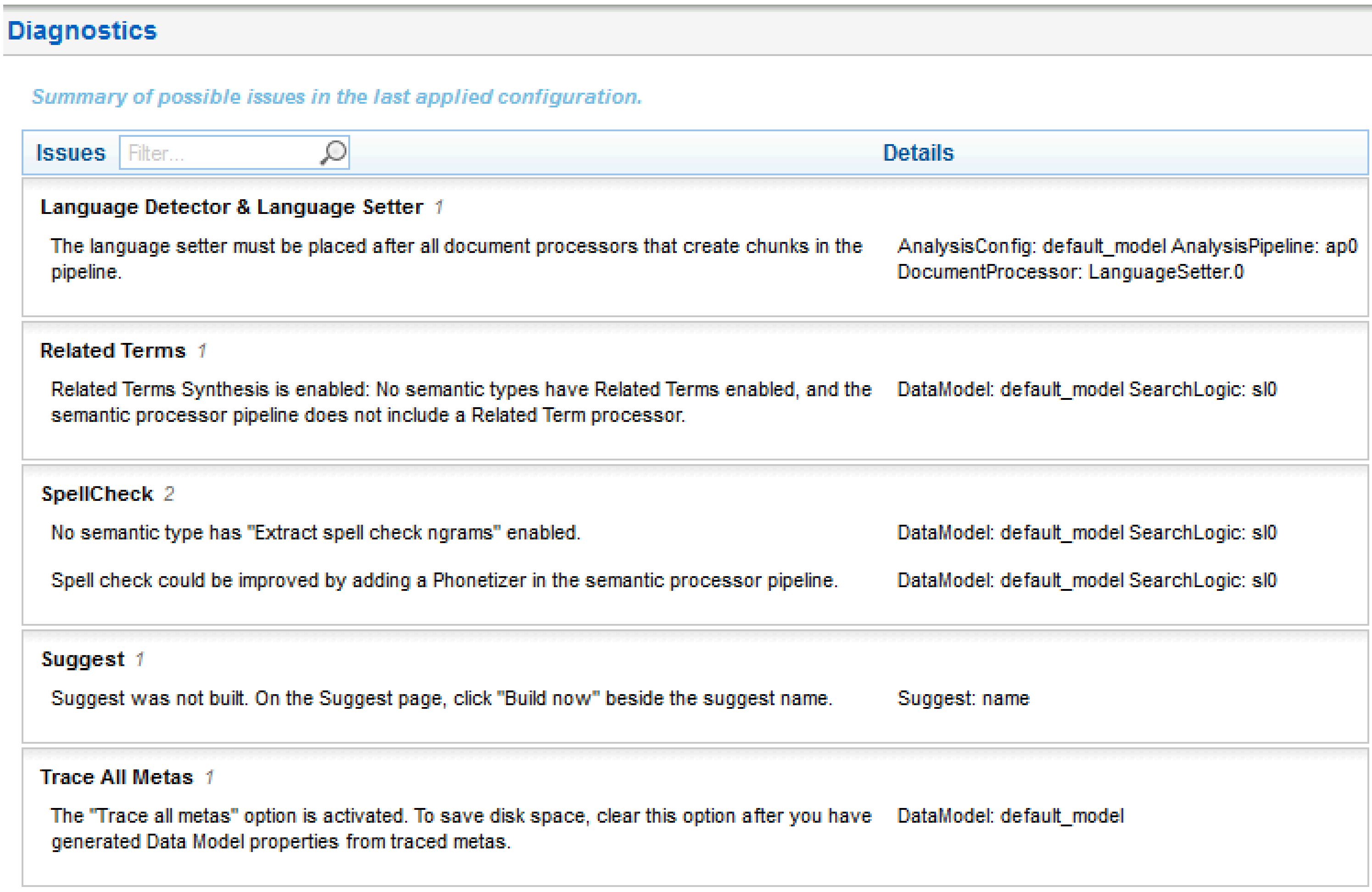
Exalead CloudView embeds monitoring features to monitor the status of your installation and tune its configuration.

While you can perform most checks and operations within the Administration Console, you can also check the system and service performances from the Monitoring Console, or use administrative tools (`cvcommand`, `cvinit`).

Identifying Configuration Issues

In the Administration Console, the **Diagnostics** menu indicates the number of outstanding issues in the last applied configuration.

You can click most of the potential issues to go directly to the related configuration page. Once you have corrected the problem and applied the configuration, the **Diagnostics** page is refreshed automatically.



Diagnostics

Summary of possible issues in the last applied configuration.

Issues	Details
Language Detector & Language Setter 1	
The language setter must be placed after all document processors that create chunks in the pipeline.	AnalysisConfig: default_model AnalysisPipeline: ap0 DocumentProcessor: LanguageSetter.0
Related Terms 1	
Related Terms Synthesis is enabled: No semantic types have Related Terms enabled, and the semantic processor pipeline does not include a Related Term processor.	DataModel: default_model SearchLogic: sl0
SpellCheck 2	
No semantic type has "Extract spell check ngrams" enabled.	DataModel: default_model SearchLogic: sl0
Spell check could be improved by adding a Phonetizer in the semantic processor pipeline.	DataModel: default_model SearchLogic: sl0
Suggest 1	
Suggest was not built. On the Suggest page, click "Build now" beside the suggest name.	Suggest: name
Trace All Metas 1	
The "Trace all metas" option is activated. To save disk space, clear this option after you have generated Data Model properties from traced metas.	DataModel: default_model

Technical Monitoring

This section details how to monitor processes and check performance issues.

Monitor Processes

Check system health and services performance

Monitor Processes

The critical processes to check are the Search server and the Index.

To...	Use...	Or connect to...
Check All Processes	`cvcmd status`	**Administration Console >Home > Processes > Host** list
Check Local Processes	`<DATADIR>\bin\ cvinit.[bat|sh] status`	**Administration Console >Home > Processes**
Check Process Logs	`<DATADIR>\run\<PROCESS NAME>\log.log`	**Administration Console >Home > Logs > Hosts** and **Processes** lists

To...	Use...	Or connect to...
Look for Restarts or Loop Failing	`http://<HOSTNAME>:<BASEPORT+11>/mami/deploy/getDeploymentStatus`	-

Check All Processes

Processes may be in the following statuses:

- Started
- Starting
- Stopped
- Dead

1. Go to `<DATADIR>/bin`.
2. Run `cvcmd[.sh] status`.

```
NAME                      STATUS PAYLOAD  PID         STARTDATE
consolidationserver-cs0     started - 7332  2018-10-07  11:15:55+0200
searchserver-ss0            started - 2260  2018-10-07  11:15:55+0200
master                      started - 9240  2018-10-07  09:48:38+0200
index6-bg0-i0               started - 9236  2018-10-07  09:48:38+0200
gateway                     started - 9250  2018-10-07  09:48:38+0200
indexingserver-bg0          started - 9239  2018-10-07  09:48:38+0200
connectors-java0            started - 23311 2018-10-07  10:49:01+0200
convert-c0                  started - 9231  2018-10-07  09:48:38+0200
crawler-exa0                started - 2213  2018-10-07  11:15:52+0200
```

Check Local Processes

1. Go to `<DATADIR>/bin`.
2. Run `<DATADIR>\bin\cvinit.bat|sh status`.

```
NAME                      STATUS   PAYLOAD PID     STARTDATE
searchserver-ss0          started -        2260     2018-10-07T11:15:55+0200
master                    started -        9240     2018-10-07T09:48:38+0200
index6-bg0-i0             started -        9236     2018-10-07T09:48:38+0200
gateway                   started -        9250     2018-10-07T09:48:38+0200
indexingserver-bg0        started -        9239     2018-10-07T09:48:38+0200
connectors-java0          started -       23311     2018-10-07T10:49:01+0200
convert-c0                started -        9231     2018-10-07T09:48:38+0200
crawler-exa0              started -        2213     2018-10-07T11:15:52+0200
```

Check Process Logs

1. Go to `<DATADIR>\run\<PROCESS NAME>\log.log`.
2. Look for `ERROR`.
3. If the log information level is not enough, set the log level to `DEBUG` in **Administration Console > Troubleshooting > Logs**.

Look for Restarts or Loop Failing

1. Go to `http://<HOSTNAME>:<BASEPORT+11>/mami/deploy/getDeploymentStatus`.
2. Look for `nbUnexpectedRestarts`, `nbConsecutiveUnexpectedRestarts`, and `loopCrashing="true"`.

```
<DeploymentStatus xmlns="exa:exa.bee.deploy.v10">
<HostStatus hostname="ngdev018.paris.exalead.com" install="cvdefault"
status="ok" architecture="amd64-linux" nbCpus="8" cpuUsage="3.0112925"
 dataDir="/data/jdoe/data_TRUNK_95767"
 installDir="/data/jdoe/cloudview-dev_trunk.dev.95767-linux-x64"
 user="jdoe" hostAgentStartupConfigVersion="17" exaHostAgentPort="16009"
 javaHostAgentPort="16027">
<ProcessStatus processName="searchserver-ss0" status="started" pid="11652"
 lastStartDate="1554377913673" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16019,16020,16000,16021,16010,16023"
 debugPort="16029" defaultPort="16019"/>
<ProcessStatus processName="master" status="started" pid="11645"
 lastStartDate="1554377913673" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16003"
 debugPort="-1" defaultPort="16003"/>
<ProcessStatus processName="index6-bg0-i0" status="started" pid="11650"
 lastStartDate="1554377913673" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16015"
 debugPort="16035" defaultPort="16015"/>
<ProcessStatus processName="gateway" status="started" pid="11644"
 lastStartDate="1554377913673" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16001,16004,16005,16006,16007,16027,16011,16008"
 debugPort="16037" defaultPort="16004"/>
<ProcessStatus processName="indexingserver-bg0" status="started" pid="11655"
 lastStartDate="1554377913674" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16012,16013,16014"
 debugPort="16031" defaultPort="16012"/>
<ProcessStatus processName="connectors-java0" status="started" pid="11648"
```

```
 lastStartDate="1554377913673" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16024,16025,16026"
 debugPort="16033" defaultPort="16024"/>
<ProcessStatus processName="convert-c0" status="started" pid="11657"
 lastStartDate="1554377913674" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16038,16017"
 debugPort="16040" defaultPort="16038"/>
<ProcessStatus processName="hostagent" status="started" pid="11609"
 lastStartDate="1554377913668" nbUnexpectedRestarts="0" loopCrashing="false"
 nbConsecutiveUnexpectedRestarts="0"
 ports="16009"
 debugPort="-1" defaultPort="16009"/>
<MemoryInfos swapSize="16777212" swapFree="16759548"
 physicalSize="12151328" physicalFree="903324"/>
</HostStatus>
</DeploymentStatus>
```

Check system health and services performance

This section describes how to use Exalead CloudView Monitoring Console and monitor performance.

Display Historic and Real-Time Data

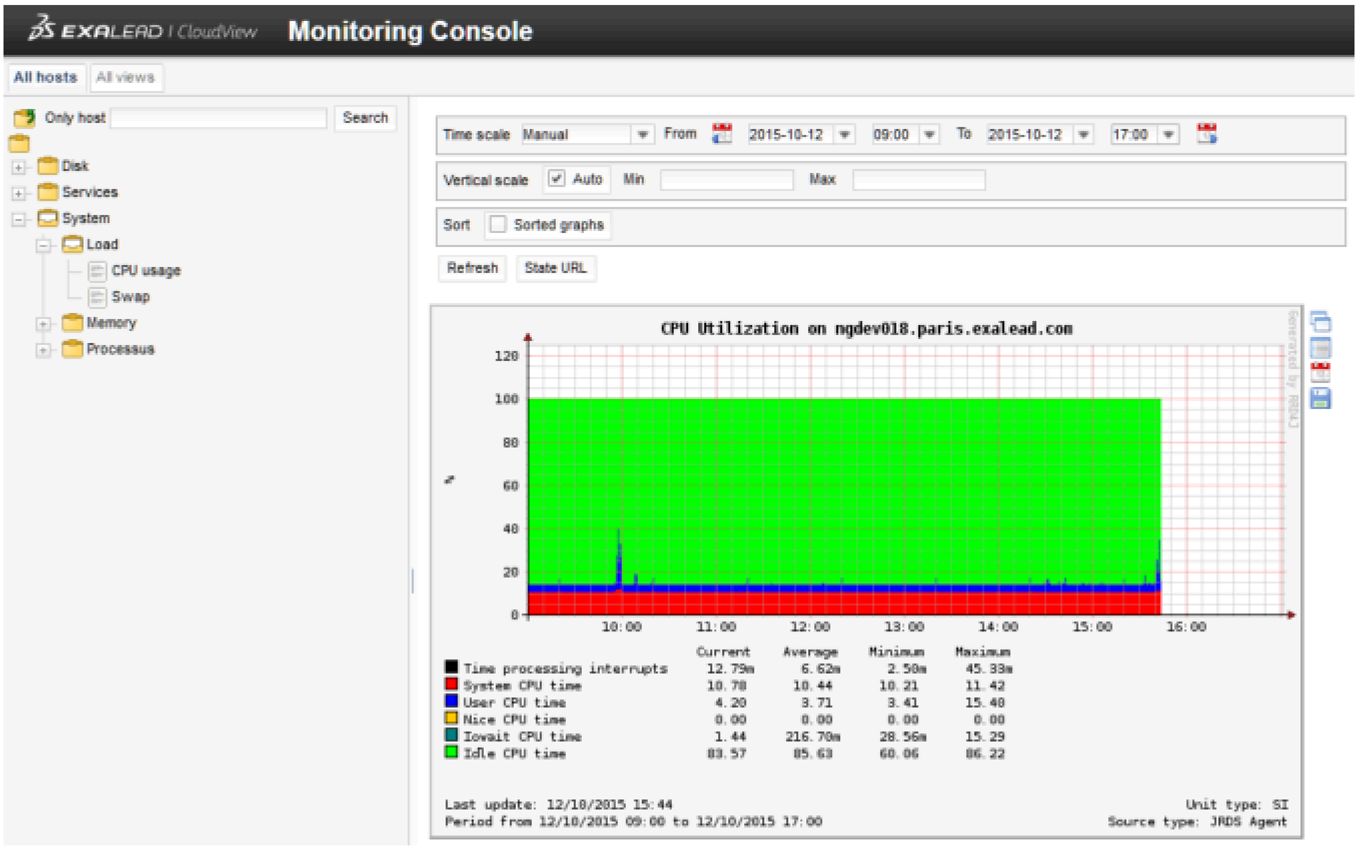
The Monitoring Console is available at: http://<HOSTNAME>:<BASEPORT+1>/perf-ui.

The Exalead CloudView Monitoring Console is based on JRDS (`http://jrds.fr/`). It provides detailed system-wide performance analysis, both historic and real-time for:

- Exalead services (search and indexing statistics, data structure size, etc.).
- System statistics (CPU load, disk and network activity, memory usage, etc.).
- Exalead process health (CPU usage, memory usage, IO activity, etc.).

1. Select the:
    - **All hosts** tab to display performances by host.
    - **All views** tab to display performances by service.
2. For each host, the tree list displays the following nodes:

- **Disk**: shows the disk I/O usage
- **Services**: shows the performances of internal services (**EXALEAD**) and the performances of the Exalead CloudView processes (**JVM**).
- **System**: shows the performances of the host machine.



Time scale: Manual | From: 2015-10-12 09:00 | To: 2015-10-12 17:00

Vertical scale: ✓ Auto | Min: | Max:

Sort: ☐ Sorted graphs

	Current	Average	Minimum	Maximum
⬛ Time processing interrupts	12.79m	6.62m	2.50m	45.33m
🟥 System CPU time	10.78	10.44	10.21	11.42
🟦 User CPU time	4.20	3.71	3.41	15.40
🟨 Nice CPU time	0.00	0.00	0.00	0.00
🟦 Iowait CPU time	1.44	216.70m	28.56m	15.29
🟩 Idle CPU time	83.57	85.63	60.06	86.22

```
Last update: 12/10/2015 15:44                                  Unit type: SI
Period from 12/10/2015 09:00 to 12/10/2015 17:00     Source type: JRDS Agent
```

3. To filter the graph views by date, specify a **Time Scale**.
4. To filter the values of the graph ordinate (vertical axis), for **Vertical** scale, clear **Auto** and specify the min and max values to display.
5. For each graph, you can:
    - Open the graph in a new pop-up window.
    - See the graph details in a new window.
    - See the graph history in a new window, with separate graphs for daily, weekly, monthly, and yearly performances.
    - Save the graph data to CSV. This can be useful to recreate the graph with another tool.

Troubleshoot Issues with Performance Monitoring

This section describes common issues that can be found through performance monitoring.

Graphs are not displayed

Problem: "When displaying the Exalead CloudView Monitoring Console, I can see the content of the tree on the left but there is no graph on the right."

Cause: Graphs may appear properly because of high CPU utilization on your server.

Solution: Check CPU on your server and reconnect to the Exalead CloudView Monitoring Console.

Identify Slow or Repetitive Queries

By default, details of search queries are logged in `<DATADIR>/run/searchserver-ss0/search-reporting/search.csv`. You can build your own monitoring based on this file.

Example:

```
#timestamp;apiclient_ip;query_logic;query_target;query_querystring;
      query_language;query_start;query_hf;answer_nmatches;answer_nhits;time_total;
      query_full;query_id;query_origin;answer_status
"2018/10/07 15:10:26.443";"192.168.206.157";"sl0";"st0";"log";"xx";"0";"10";"7";"7";
"549803";"q=log";"8";
"";"ok"
"2018/10/07 15:10:40.137";"192.168.206.157";"sl0";"st0";"test";"xx";"0";"10";"10";
"10";"375940";
"q=test";"10";"";"ok"
"2018/10/07 15:10:55.535";"192.168.206.157";"sl0";"st0";"troubleshooting";"xx";"0";
"10";"7";"7";"344747";
"q=troubleshooting";"12";"";"ok"
"2018/10/07 15:11:05.523";"192.168.206.157";"sl0";"st0";"rollback";"xx";"0";"10";
"2";"2";"159574";
"q=rollback";"14";"";"ok"
```

From the **Administration Console > Reporting**, you can:

- Change the default format of the output file
- Add/remove fields written to file
- Define rotation parameters

For more information, see in the Exalead CloudView Configuration Guide.

Build Your Own Monitoring

You can retrieve all the probes monitored in the Monitoring Console in your own monitoring system through JMX.

What Is JMX?

JMX is a technology used to monitor Java applications. JMX uses objects called MBeans to expose data and resources from Exalead CloudView.

You need to use a JMX client to retrieve probes from Exalead CloudView. In the example below, we use the JConsole included within your JDK.

Activate JMX

JMX is no longer activated by default on the gateway since Exalead CloudView 2020x.R1.

To allow JMX, you need to add these extra args to the `DeploymentInternal.xml` gateway process in the `<deploy:args>` section:

```
<bee:StringValue value="-Dcom.sun.management.jmxremote" />
<bee:StringValue value="-Dcom.sun.management.jmxremote.port=PORT_OUT_OF_CV_PORT_RANGE
<bee:StringValue value="-Dcom.sun.management.jmxremote.authenticate=false" />
<bee:StringValue value="-Dcom.sun.management.jmxremote.ssl=false" />
```

Retrieve Probes

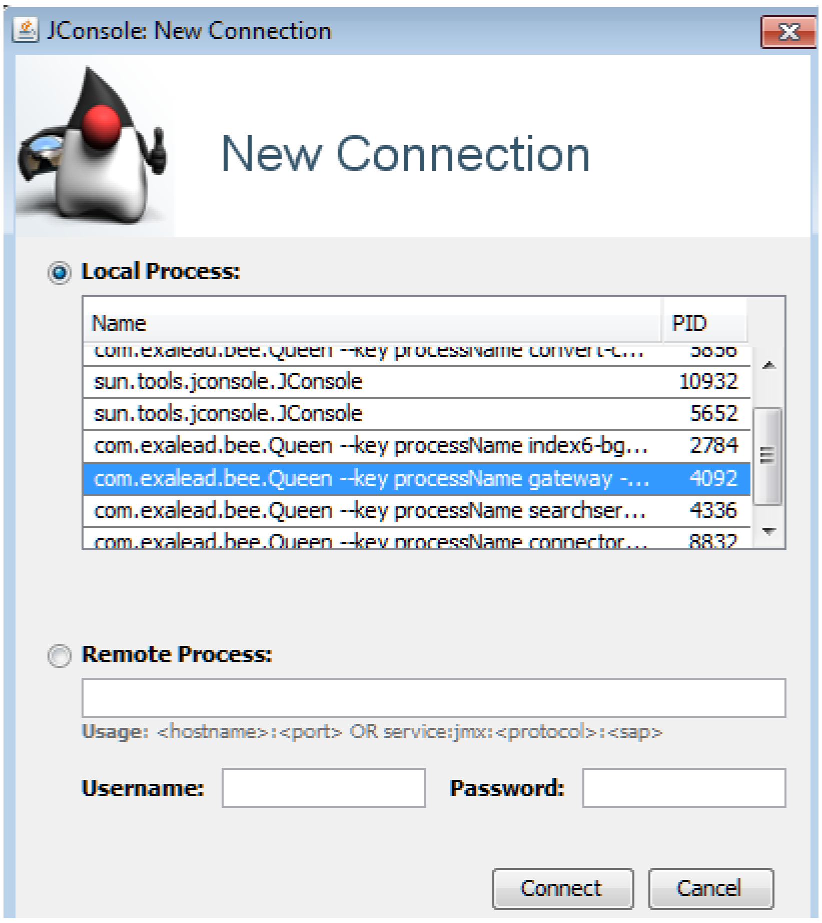
The JMX port of the gateway retrieves probes in `MBeans/jrds/Management/Operations/getLastValues(hostname, probename)`, where:

- `hostname` is the hostname specified in `Deployment.xml`.
- `probename` is the name of the probe (that is, the file name without the `.RRD` extension).
  You can find names of available probes in `<DATADIR>/perfmonitoring/probe/<HOSTNAME>/`.

Example: Retrieve Disk Usage Using JConsole

The procedure below only applies to JConsole on a specific probe. You may use other JMX clients and probe names to build your own monitoring.

1. Run the JConsole application.
2. Select the `gateway` process and click **Connect**.


JConsole: New Connection

New Connection

Local Process:

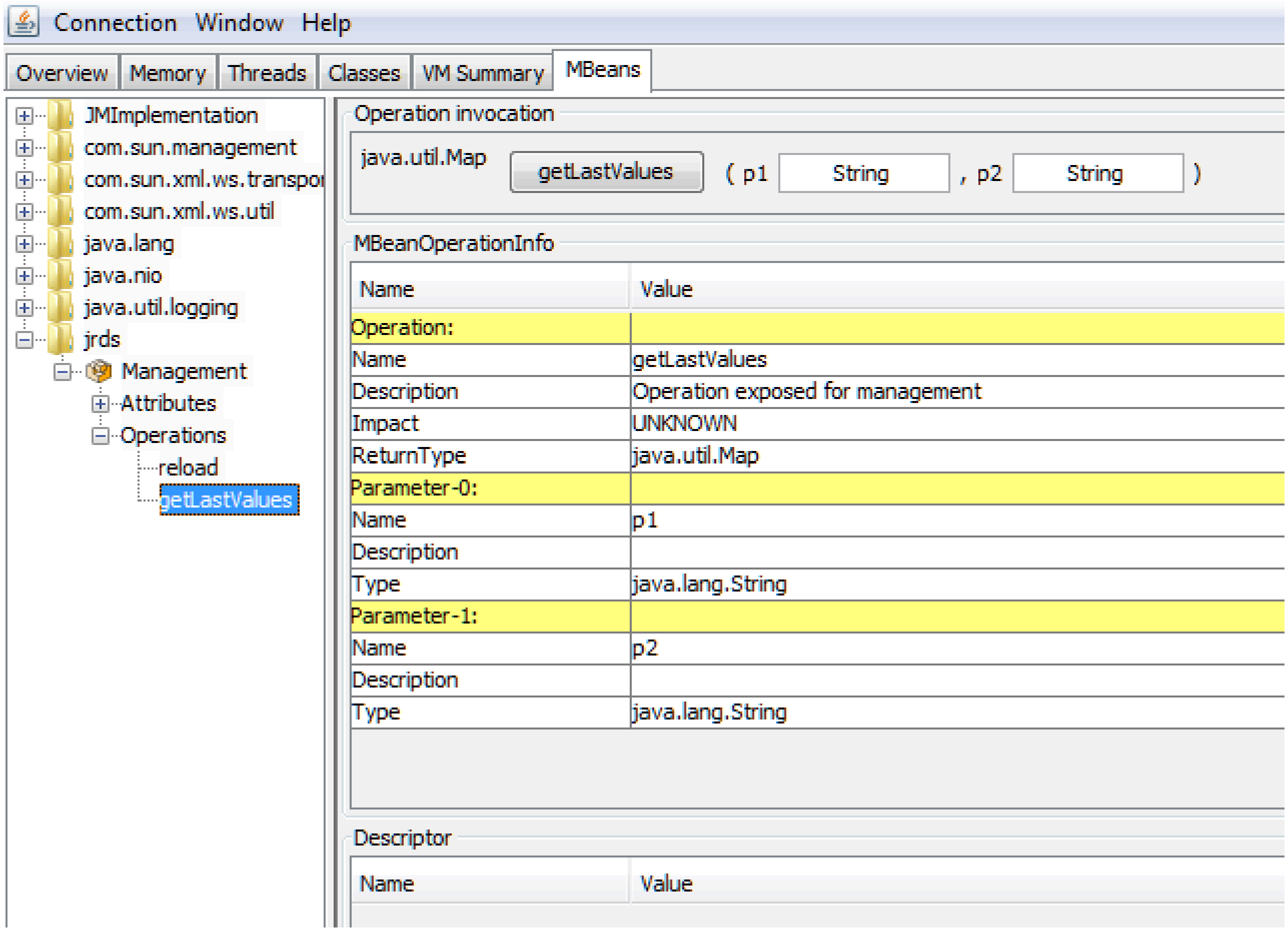
Name	PID
com.exalead.bee.Queen --key processName convert-c...	5836
sun.tools.jconsole.JConsole	10932
sun.tools.jconsole.JConsole	5652
com.exalead.bee.Queen --key processName index6-bg...	2784
com.exalead.bee.Queen --key processName gateway -...	4092
com.exalead.bee.Queen --key processName searchser...	4336
com.exalead.bee.Queen --key processName connector...	8832

Remote Process:

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

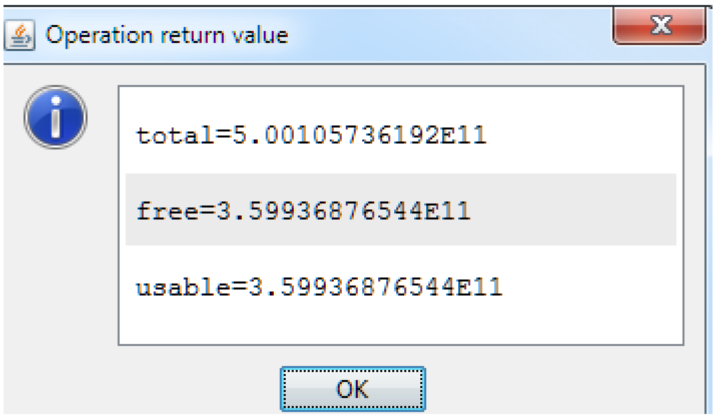
Username: Password:

3. Once connected, select the **MBeans** tab.
4. Select **jrds > Management > Operations > getLastValues**. The following screen displays.



Name	Value
Operation:	
Name	getLastValues
Description	Operation exposed for management
Impact	UNKNOWN
ReturnType	java.util.Map
Parameter-0:	
Name	p1
Description	
Type	java.lang.String
Parameter-1:	
Name	p2
Description	
Type	java.lang.String

5. Enter:
   - The hostname in the **p1** field.
   - The name of the probe in the **p2** field. For this example, enter `du` to display disk usage.
6. Click **getLastValues**. A pop-up window displays the values of the probe:



```
total=5.00105736192E11
free=3.59936876544E11
usable=3.59936876544E11
```

Functional Monitoring

This section describes how to check connectors, indexing and search.

Check Connectors

Check Indexing

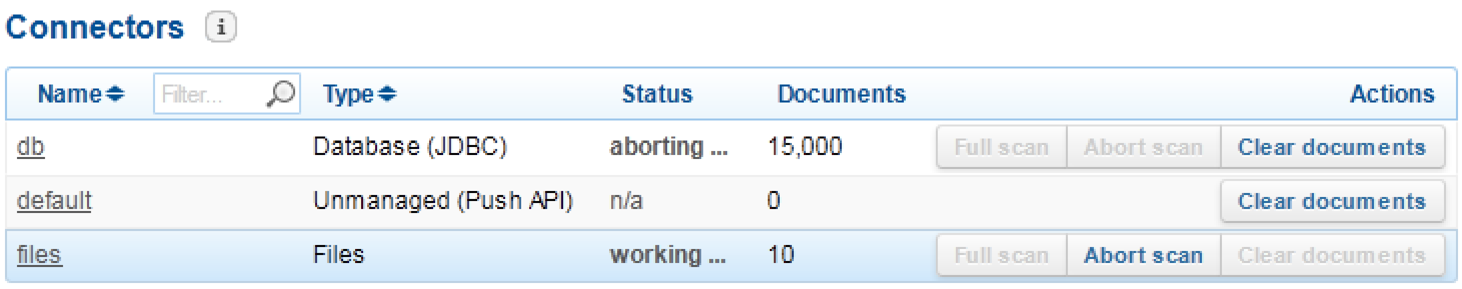
Check Search

Check Connectors

This section explains how to check the Exalead CloudView connectors.

Check Overall Connector Status

In the Administration Console, the **Home > Connectors** section provides status information and displays the number of scanned documents.



Connectors

Name	Type	Status	Documents	Actions
db	Database (JDBC)	aborting ...	15,000	Full scan / Abort scan / Clear documents
default	Unmanaged (Push API)	n/a	0	Clear documents
files	Files	working ...	10	Full scan / Abort scan / Clear documents

Check Actions on Documents and Objects

For a more detailed connector status, you can check actions performed on documents.

To check actions on documents, connect to `http://<HOSTNAME>:<BASEPORT+11>/mami/connect/getConnectorsStatus` and for each connector, look for:

- Number of deleted documents, `totalDeletes`.
- Number of replaced documents, `totalReplaces`.
- Number of deleted objects, `deletedObjects`.
- Number of pushed objects, `pushedObjects`.
- Number of scanned objects, `scannedObjects`.

Example:

```
<connect:ConnectorsStatus>
<connect:ConnectorStatus runtime="java"
 classId="com.exalead.papi.connectors.filesystem.FilesystemConnector"
 status="idle" connectorServer="java0" managed="true" connectorName="doc">
<connect:PerBuildGroupStatus totalPartialUpdates="0" totalFailedDeletes="0"
 totalDeletes="11" totalReplaces="0"
 totalAdds="22" activeDocumentsCount="11" buildGroup="bg0"/>
<connect:previousScan>
<connect:ScanStatus deletedObjects="0" pushedObjects="11" scannedObjects="11"
 endTime="1381137322180"
 startTime="1381137319607" time="2573" aborted="false">
<connect:specificMeasures/>
</connect:ScanStatus>
</connect:previousScan>
</connect:ConnectorStatus>
<connect:ConnectorStatus managed="false"
 connectorName="default"><PerBuildGroupStatus totalPartialUpdates="0"
 totalFailedDeletes="0" totalDeletes="0" totalReplaces="0" totalAdds="0"
 activeDocumentsCount="0"
 buildGroup="bg0"/>
</connect:ConnectorStatus>
<connect:ConnectorStatus status="unknown" managed="false"
 connectorName="test">
<connect:PerBuildGroupStatus totalPartialUpdates="0" totalFailedDeletes="0"
 totalDeletes="0" totalReplaces="0"
 totalAdds="2" activeDocumentsCount="2" buildGroup="bg0"/>
</connect:ConnectorStatus>
</connect:ConnectorsStatus>
```

Check Indexing

This section describes how to check build groups, index size and the number of documents.

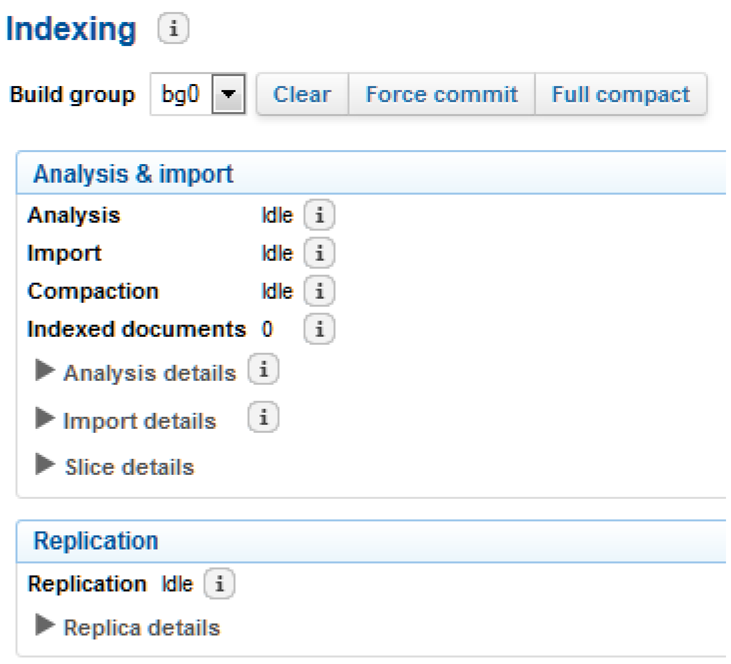
Check Overall Build Group Status

Check Index Size and Number of Files in the Index

Check Number of Documents and Commit Dates

Check Overall Build Group Status

In the Administration Console, the **Home > Indexing > Build group** section provides information related to analysis, import, replication, and the number of indexed documents.

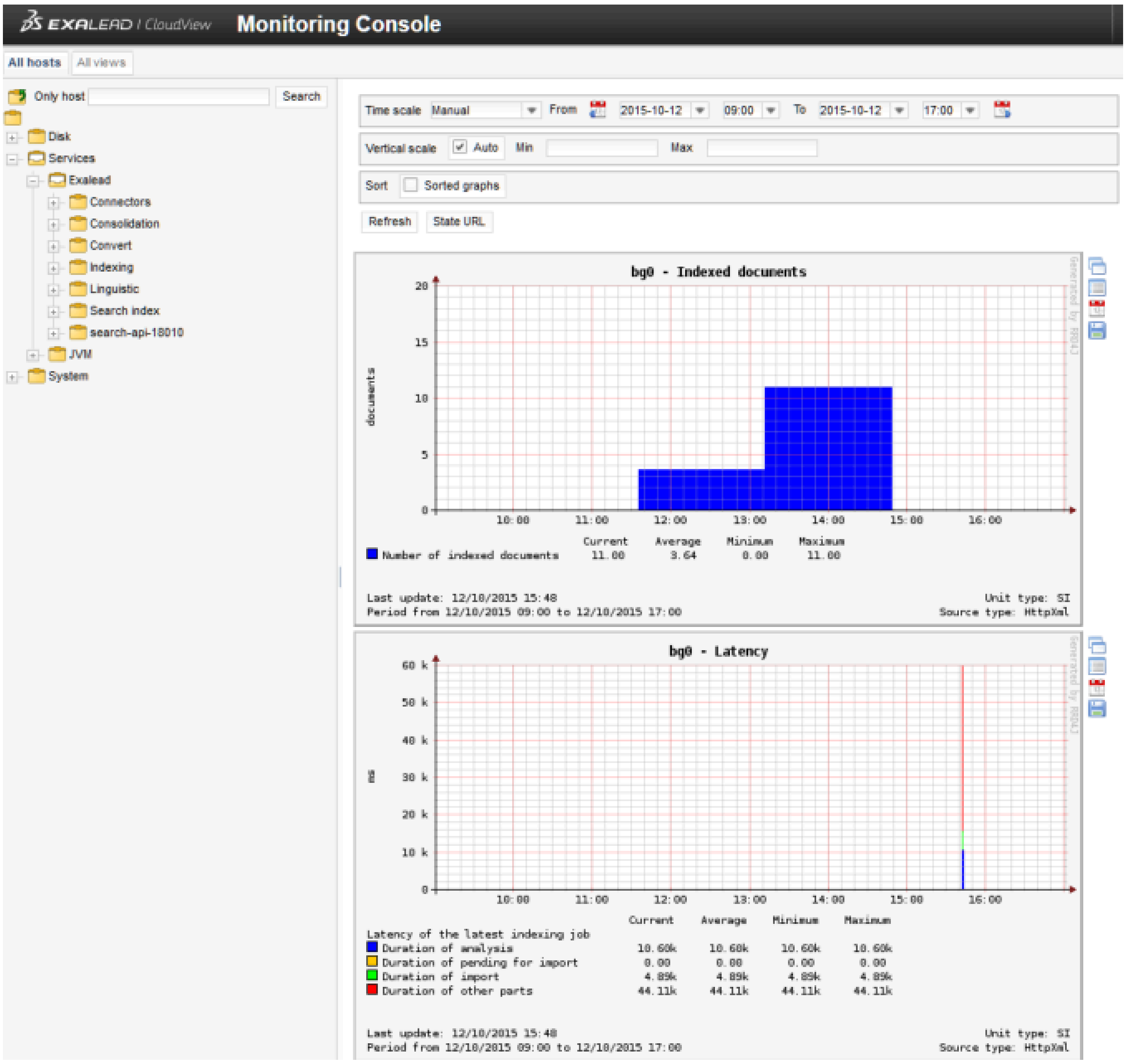


Check Index Size and Number of Files in the Index

For each build group/slice, you can display several graphs including index size, number of files in the index, indexed documents etc.

1. Open the Exalead CloudView Monitoring Console at `http://<HOSTNAME>:<BASEPORT+1>/perf-ui`.
2. Expand **Services > Exalead > Indexing**.

   Several graphs are displayed per build group/slice:



bg0 - Indexed documents

```
                                Current   Average   Minimum   Maximum
Number of indexed documents      11.00      3.64      0.00     11.00

Last update: 12/10/2015 15:48                                   Unit type: SI
Period from 12/10/2015 09:00 to 12/10/2015 17:00           Source type: HttpXml
```

bg0 - Latency

```
                                       Current   Average   Minimum   Maximum
Latency of the latest indexing job
Duration of analysis                    10.60k    10.60k    10.60k    10.60k
Duration of pending for import           0.00      0.00      0.00      0.00
Duration of import                       4.89k     4.89k     4.89k     4.89k
Duration of other parts                 44.11k    44.11k    44.11k    44.11k

Last update: 12/10/2015 15:48                                   Unit type: SI
Period from 12/10/2015 09:00 to 12/10/2015 17:00           Source type: HttpXml
```

Check Number of Documents and Commit Dates

For a more detailed status of each build group, you can check that:

- The number of documents corresponds to the estimated range of expected documents.
- Commits are made on time.

Note: You can also display additional information, for example, RAM usage, deletions, updates etc.

1. Go to `http://<HOSTNAME>:<BASEPORT+11>/mami/indexing/getBuildGroupStatus?buildGroup=<BUILD_GROUP_CODE>`.
2. Look for:

- The number of documents in this build group `<IndexSliceBuilderStatus ndocs=...>`.
- The last commit date in `<IndexSliceInstanceStatus lastCommit=...>`.

```
<BuildGroupStatus
  xmlns="exa:com.exalead.mercury.mami.indexing.v10"
  xmlns:ns2="exa:com.exalead.indexing.analysis.v10"
  xmlns:ns3="exa:exa.bee"
  xmlns:ns4="exa:com.exalead.search.v30"
  xmlns:ns5="exa:com.exalead.ndoc.v10">
<PushServerStatus dihCompacting="false" valid="true" enabled="true" />
- <IndexingStatus valid="true">
- <AnalysisStatus ramUsageLimit="2147483648" ramUsage="32629960"
 elapsedMS="0" currentAttempt="1"
 totalBytes="0" totalTasks="0" totalDeletions="0"
 totalPartialUpdates="0" totalExistingDocumentAdds="0"
 totalNewDocumentAdds="0" processing="false" enabled="true">
- <newDocumentAdds>
  </bytes>
  </ImportStatus>
  </IndexingStatus>
- <IndexBuilderStatus replicating="false" compacting="false"
  totalDocs="11" complete="true">
- <IndexSliceBuilderStatus ndocs="11" maxDid="535" nfree="0"
 lastCommit="1381156473695"
 serial="1381156472119" compacting="false" importing="false"
 valid="true" indexSlice="0">
  <CompactingStatus progress="0" globalProgress="0" />
  </IndexSliceBuilderStatus>
  </IndexBuilderStatus>
  <ThumbnailsStatus deletedThumbs="0" failedThumbs="0"
  computedThumbs="0" ignoredDocuments="0"
  skippedDocuments="0" processedDocuments="0" processedTasks="0"
  valid="false" />
  <IndexSliceInstanceStatus lastCommit="1381156473000"
  replicationPaused="false"
  replicationAttached="true" replicationSizeToDownload="0"
  replicationFilesToDownload="0"
  replicating="false" status="ok" serial="1381156472119"
  valid="true" install="cvdefault"
  host="ngdev018.paris.exalead.com" sliceInstance="i0" indexSlice="0" />
   <DocumentCacheRepushStatus valid="true" repushedDocuments="0"
   startTime="0" running="false" />
</BuildGroupStatus>
```

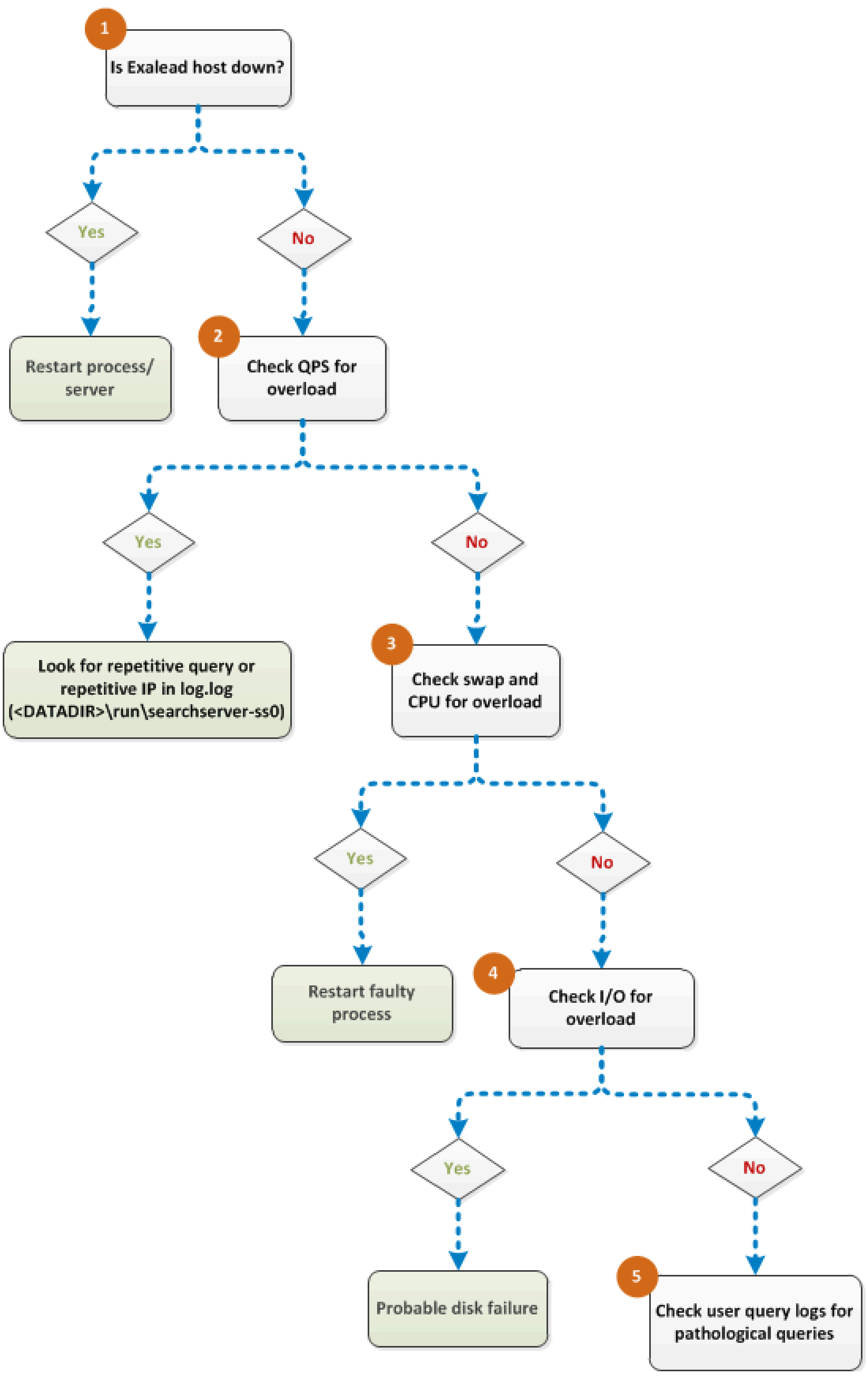
Check Search

This section is dedicated to the monitoring of search server health and performance.

If you encounter index crash or unexpected search results, see "Troubleshooting Document Analysis" in the Exalead CloudView Configuration Guide.

Troubleshoot Search Performance

If you encounter performance issues at search time, follow the steps below to find out why:


1
Is Exalead host down?
Yes
No
Restart process/ server
2
Check QPS for overload
Yes
No
Look for repetitive query or repetitive IP in log.log (<DATADIR>\run\searchserver-ss0)
3
Check swap and CPU for overload
Yes
No
Restart faulty process
4
Check I/O for overload
Yes
No
Probable disk failure
5
Check user query logs for pathological queries

1. Check if Exalead CloudView host is down using `cvcmd`. See Check All Processes.
2. Check QPS (query per second) to identify overload using the Monitoring Console. See Display Historic and Real-Time Data.

3. Check swap and CPU to identify overload using the Monitoring Console. See Display Historic and Real-Time Data.
4. Check I/O to identify overload using the Monitoring Console. See Display Historic and Real-Time Data.
5. Check user query logs for pathological queries using:
   - Log file in DEBUG in `<DATADIR>\run\searchserver-ss0`. See Check Process Logs.
   - Details of search queries logged in `/<DATADIR>/run/searchserver-ss0/search-reporting/search.csv`. See Identify Slow or Repetitive Queries.

Monitor Search Server Health with Load Balancers

To enable external load balancers to monitor the health of Exalead CloudView search servers, specify an `isAlive` query. Then run this query at regular intervals to verify that its target indexes are still available. When the query fails, the status changes to "not alive".

You can also configure load balancers to check HTTP return error code:

- 200 = alive
- Any other value = not alive

Monitor Both the Search API and the Mashup UI

This checks that both the Search API and the Mashup UI are working correctly, by sending a lightweight query at regular intervals.

1. Go to Mashup Builder and select **Application > Application properties**.
2. In **isAlive query**, enter your test query. For example, `/page/<page>?q=gizmo` (replace `gizmo` with your own query term).

   Important: This query is regularly sent to all indexes associated with this application. To avoid impacting performance, specify a simple query that returns few, if any, matching documents.

3. Click **Save** and then apply your changes.
4. On your load balancer, enter this URL to monitor search server health: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/isAlive`.

Monitor the Search API Only

If your deployment uses a custom search application created through the Search API, instead of the Mashup UI, modify the `SearchCommand` to include one or more `isAlive` queries. You need to push a document that matches this query.

1. Open `<DATADIR>/config/SearchAPI.xml`.

2. Modify `<SearchCommand>` to include the `<isAliveQueries>` nested element, as shown below (replace `foobar` with your query term).

   ```
   <SearchCommand isAliveAsynchronousDelayS="5" isAliveSynchronous="true"
    enableSOAP="false" maxConcurrentQueries="10" targetReporting="search-api" default
    defaultLogic="sl0" base="/search-api">
   <isAliveQueries>
   <ns2:StringValue value="q=foobar"/>
   </isAliveQueries>
   </SearchCommand>
   ```

   Important: This query is regularly sent to all search servers associated with this application. To avoid impacting performance, specify a simple query that returns few, if any, matching documents.

3. (Optional) Add additional `StringValue` elements for additional queries.
4. Apply changes manually. See "Installing slave hosts".
5. On your load balancer, enter this URL to monitor search server health: `http://<HOSTNAME>:<BASEPORT+10>/search-api/isAlive`.

Display Query Statistics Using a Simple Search Request

This page describes how to get query statistics from a simple search request using the Search API.

Display Query Statistics

You can submit a simple search request (word) to: `http://<HOSTNAME>:<BASEPORT>/search-api/search?q=<SEARCH_REQUEST>`.

In the `<stats>` node, at the end of the page, you get the result of the query and information on query processing time and slices.

Example:

```
<Stats status="ok" queueTime="0" queryProcessingElapsedTime="5151"
 queryProcessingCPUTime="3302"
 queryExecElapsedTime="3139"
 queryExecIndexCPUTime="768"
 queryExecSearcherCPUTime="1951"
 synthesisAndFullHitsElapsedTime="177566"
 synthesisIndexCPUTime="836"
 synthesisSearcherCPUTime="59"
 fullHitsIndexCPUTime="3818"
 fullHitsSearcherCPUTime="0"
 totalProcessingTime="180812" />
 - <slicesInfo>
  <SliceInfo instance="i0" slice="0" internalGeneration="117"
```

```
  lastCommit="1381151879238"
  lastJobId="1381151878907" />
   </slicesInfo>
```

Attribute Description

All attributes are expressed in microseconds.

Stats attribute	Description
queueTime	When the maximum number of queries processing in parallel is reached, next queries are queued until the first ones finish. This parameter gives the time the query has waited for a free slot.
queryProcessingElapsedTime queryProcessingCPUTime	Processing time is the time spent in the search server, between receiving the query from the client, and sending it to the indexes. This includes parsing, and eventually expanding the query according to the search query processors. Elapsed time and CPU time must be similar, because processing is single-threaded and mostly CPU-bound. Expansions however can use the dictionary and wait for disk IO.
queryExecElapsedTime queryExecIndexCPUTime queryExecSearcherCPUTime	This is the time spent collecting matching document IDs in the indexes. The Index CPU Time is the CPU spent in the indexes (all slices in parallel), and searcher CPU Time is the time spent in the search server (much lower as it only has to sort the hits).
synthesisAndFullHitsElapsedT	This is the time spent making the synthesis from the top hits (either all, or top 10,000 usually). This makes lots of requests to ram-based structures in the indexes containing the facets seach document matches. The full hits consist only of retrieving the metas from the attribute groups for the top 10 results.
synthesisIndexCPUTime synthesisSearcherCPUTime	CPU Time spent in searcher and indexes for the synthesis. Indexes spend a lot of time processing ram-based structures to find the facets for each document. The searcher just stores and updates the totals, and then sorts them to find the top facets.

Stats attribute	Description
`fullHitsIndexCPUTime` `fullHitsSearcherCPUTime`	This is the CPU time spent to retrieve the metas for the top 10 documents from the attribute groups. CPU time is usually very low because these accesses are easy, or maybe you had no results?
`totalProcessingTime`	This is the total elapsed time to process the query and return the results, which match the sum of processing, `queryExec` and `synthesisAndFullHits` elapsed times. Do not add CPU times, as they include many CPUs in parallel so they can obviously be higher than the elapsed time.

Configuring Logs

This section describes how to configure logs.

Where Are the Process Logs?

Change Log Location

Rotate and Purge Logs

Where Are the Process Logs?

Logs are available for each process. They are written to `<DATADIR>\run\<PROCESS NAME>\log.log` and can be displayed from **Administration Console > Logs.**

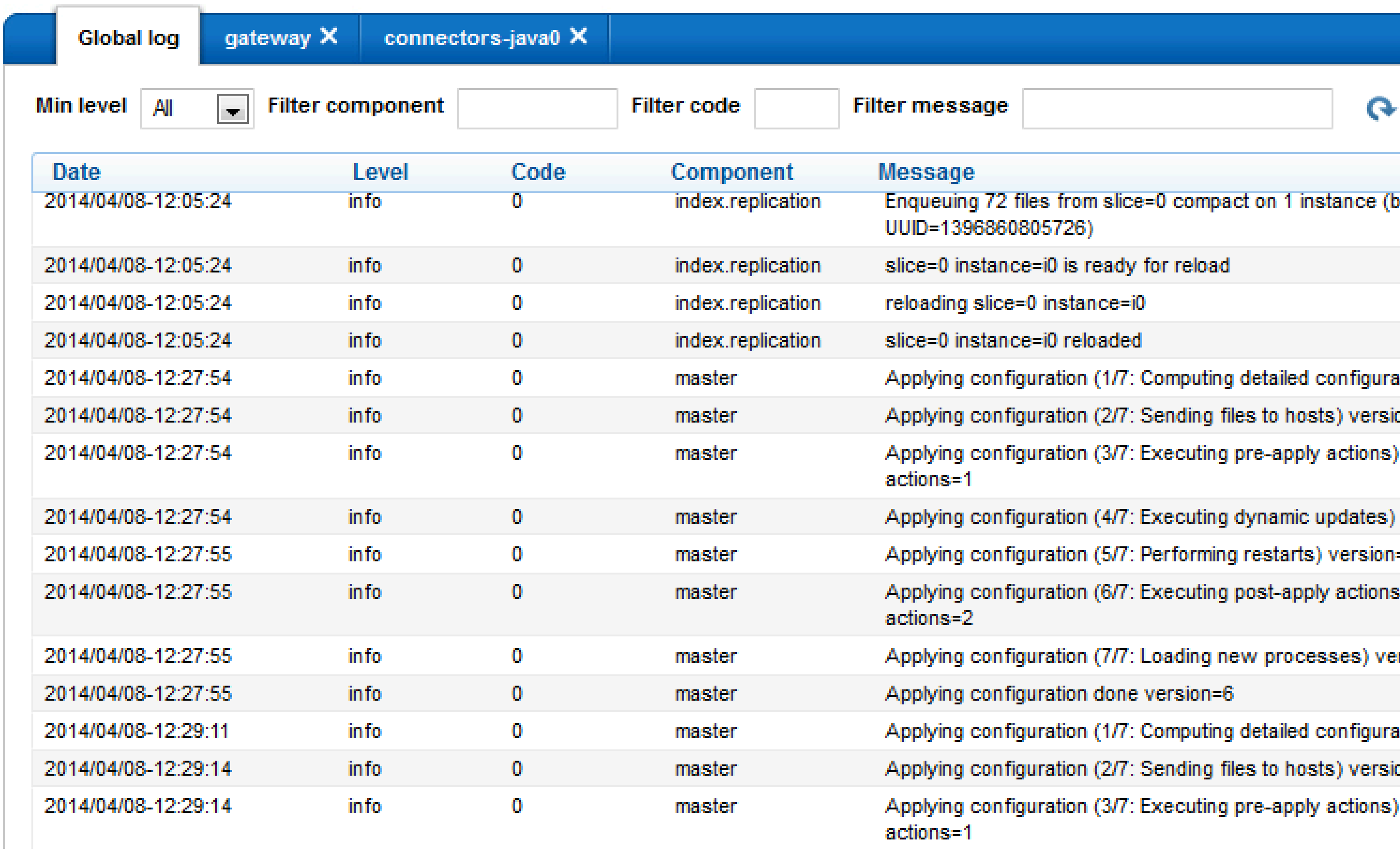
A global log file gathering all process logs is written to `<DATADIR>\run\global.log`.

How to Find Errors?

Search for `ERROR` and `WARN` to detect errors in logs.

How to Change the Log Level?

Use the **Min level** field to filter the log level displayed in **Administration Console > Logs**:



Date	Level	Code	Component	Message
2014/04/08-12:05:24	info	0	index.replication	Enqueuing 72 files from slice=0 compact on 1 instance (b UUID=1396860805726)
2014/04/08-12:05:24	info	0	index.replication	slice=0 instance=i0 is ready for reload
2014/04/08-12:05:24	info	0	index.replication	reloading slice=0 instance=i0
2014/04/08-12:05:24	info	0	index.replication	slice=0 instance=i0 reloaded
2014/04/08-12:27:54	info	0	master	Applying configuration (1/7: Computing detailed configura
2014/04/08-12:27:54	info	0	master	Applying configuration (2/7: Sending files to hosts) versio
2014/04/08-12:27:54	info	0	master	Applying configuration (3/7: Executing pre-apply actions) actions=1
2014/04/08-12:27:54	info	0	master	Applying configuration (4/7: Executing dynamic updates)
2014/04/08-12:27:55	info	0	master	Applying configuration (5/7: Performing restarts) version=
2014/04/08-12:27:55	info	0	master	Applying configuration (6/7: Executing post-apply actions actions=2
2014/04/08-12:27:55	info	0	master	Applying configuration (7/7: Loading new processes) ver
2014/04/08-12:27:55	info	0	master	Applying configuration done version=6
2014/04/08-12:29:11	info	0	master	Applying configuration (1/7: Computing detailed configura
2014/04/08-12:29:14	info	0	master	Applying configuration (2/7: Sending files to hosts) versio
2014/04/08-12:29:14	info	0	master	Applying configuration (3/7: Executing pre-apply actions) actions=1

Recommendation: Use the `INFO` level for production environment.

Change Log Location

You may want to move the logs to another partition than the data so that a full filesystem or heavy logging does not impact the rest of the product functions.

1. Edit the file `<DATADIR>/bin/ngstart.env`.
2. Update the value of the `NGRUNDIR` variable.

   Your original `ngstart.env` is as follows:

   ```
   ...
   NGRUNDIR=C:\CloudView\data\run
   ...
   ```
3. Restart the product.

Rotate and Purge Logs

You can set log parameters in the API Console to manage the thresholds and rotation behavior.

Set Log Rotation and Purge

1. In the API Console, select **Manage**.
2. Set the log rotation and purge parameters:
   a. **Select MAMI master**.
   b. Under **Configuration**, select the `setLoggingConfig` method.
   c. Edit the parameters:

`logRotationMaxSize`	Rotate log files when bigger than the specified number of kilobytes.
`logPurgeMaxSizeKB`	Purge log files when their total size is above the specified number.
`logExpirationDays`	Purge log files when older than the specified number of days.
`logRotationCronExp`	Trigger log rotation with a custom cron expression.

3. Click **Save**.
4. Click **Apply** to apply the parameters.

Set Log Rotation Per Host

The logs are rotated by default every 24 hours using a cron expression. You can specify rotate the process log files manually using the `RotateLog` method at a different frequency.

1. In the API Console, click **Manage**.
2. Set the log rotation and purge parameters:
   a. **Select MAMI master**.
   b. Under **Operation**, select the `rotateLog` method and edit it.
   c. Click **Send**.

   The log files are archived. See the use cases below.

   By default, the `rotateLog` message is the following.

```
<RotateLog xmlns="exa:com.exalead.mercury.mami.master.v10" host="$HOST_NAME"
 install="$INSTALL_NAME"
 process="$PROCESS_NAME"/>
```

Use Cases

1. Use case 1 - To rotate all log files, edit the `RotateLog` as follows.

```
<RotateLog xmlns="exa:com.exalead.mercury.mami.master.v10" host="$HOST_NAME"
 install="$INSTALL_NAME"
 process="$PROCESS_NAME"/>
```

2. Use case 2 - To rotate the log files for the `localhost`, edit the `RotateLog` as follows.

```
<RotateLog xmlns="exa:com.exalead.mercury.mami.master.v10"
```

```
install="cvdefault" host="localhost"/>
```

3. Use case 3 - To rotate the log files for the process `master`, edit the `RotateLog` as follows.

```
<RotateLog xmlns="exa:com.exalead.mercury.mami.master.v10"
 install="cvdefault" host="localhost" process="master"/>
```

Configuring Email Alerts

This section describes how to configure e-mails alerts.

Events Generating Alerts

Enable Alerts If Not Defined at Setup

Events Generating Alerts

Emails alerts about critical events are sent automatically if you entered your email address at setup time.

If you did not specify an email address, see Enable Alerts If Not Defined at Setup.

Mail about...	When...	Frequency
Licensing	• the license is about to expire or has expired • tokens are about to run out or have run out	1 mail max/ day
Processes	at least one process in dead/loop-crashing/ starting status	1 mail max/ hour
Indexing	at least one unavailable/invalid/failing build group status	1 mail max/ hour
Search	at least one dead/unavailable search API command	1 mail max/ hour

Enable Alerts If Not Defined at Setup

If you did not enter your email address at setup time, follow the procedure below.

1. In the Administration Console, go to **Search > Reporting**.
2. In **Notifications**, select **Enable**.
3. Select the events for which you want to receive an alert:
   - **For license**. See Configure Email Alerts When Running Low on Tokens for more details.
   - **For processes**

    - **For indexing**
    - **For search**
4. Enter the sender and recipient emails.
5. Expand **SMTP server settings** and set the properties for outgoing mail:
    a. **Host**: SMTP server host.
    b. **Port**: SMTP server port.
    c. **Username**: admin user name on the SMTP server.
    d. **Password**: admin password on the SMTP server.
    e. **Use TLS**: activates Transport Layer Security encryption.
6. Click **Apply**.

Fixing Out of Memory Issues

If you encounter one of the following issues for a process in the log, the memory setting may be too low:

- `java.lang.OutOfMemoryError: Java heap space`
- `java.lang.OutOfMemoryError: PermGen space`

You can increase memory allocation for any Exalead CloudView process in `<DATADIR>/config/DeploymentInternal.xml`. By default, the following processes are already available in the file:

- The search server.
- The indexing server.
- All Java processes.

Note: Exalead CloudView processes are C++ and Java-based. Memory is mainly consumed by the C++ part. Changing memory setting as explained below does not impact memory allocation in the C++ part.

Solve java.lang.OutOfMemoryError: Java Heap Space

To solve this issue, increase the maximum heap size (`-Xmx` parameter).

1. Go to `<DATADIR>/config`.
2. Edit `DeploymentInternal.xml` and look for the process in which the issue occurred:
    - `searchserver`
    - `indexingserver`

- `java` (for all processes)

3. Increase `-Xmx` by 50%.
4. In the Administration Console Top Bar, click **Apply**.

   Note: If you cannot access the Administration Console, rebuild the configuration by running `<DATADIR>/bin/buildgct`.

Solve java.lang.OutOfMemoryError: PermGen Space

To solve this issue, specify the size of permanent generation (`-XX:MaxPermSize` parameter).

1. Go to `<DATADIR>/config`.
2. Edit `DeploymentInternal.xml` and look for the process in which the issue occurred:
   - `searchserver`
   - `indexingserver`
   - `java` (for all processes)
3. Increase `-XX:MaxPermSize` by 50%.
4. In the Administration Console Top Bar, click **Apply**.

   Note: If you cannot access the Administration Console, rebuild the configuration by running `<DATADIR>/bin/buildgct`.

Sending System Reporting

You can use the system report **ONLY** if requested by technical support. You can follow the wizard under the **Help > Contact support** to generate the standard system report (CVDiag) or from the command line.

The system report embeds:

- Logs of the process of the master and slaves.
- The product configuration + history of the configurations used.
- Machine-specific data (CPU, memory, disks).
- List of files in DATADIR.
- List of installed hotfixes.
- Lists of queries run with stats.
- List of indexing errors.

- Data for generating graphs visible in monitoring console.

Note: Confidential information (for example, user login, password) and index data do not appear in the system report.

Create a System Report Using the Administration Console

1. Go to **Help > Create system report**.
2. Click **Run system report**.

   This opens a **Run system report** window. It is best to keep the default settings:

   a. **Collect data from all hosts**: clear this if you only want the master host.

   b. **Collect process logs**: use the defaults unless you need to set limits for the process logs.

   c. **Max log size per process (MB)**

   d. **Max log days**

   e. **Max GCT version days**

   f. **Logs from only these processes**: enter a comma-separated list of process names to collect logs for. If left blank, logs from all processes are collected.

   g. **Collect core dumps**: by default the core dump is limited to one per process.

   h. **Collect managed stacks**

   i. **Collect native stacks**: only select this if instructed by Technical support.
3. Click **Run** to generate the report.

   You can either download or upload to an Exalead server.

Create a System Report from the Command Line

1. Go to `<DATADIR>/bin`.
2. Run `cvdiag`. You can find the generated ZIP file in `<DATADIR>/cvdiags/cvdiag_DATE-TIME.zip`.

   Note: You can specify another output directory and file name using the following command: `cvdiag /MY_PATH/my_cvdiag.zip`.

Upload to EXALEAD

The report upload does not normally require any additional configuration. Upload options are available for specific proxy connections.

1. Configure the following:

   a. **HTTP proxy host** : leave blank for a direct internet connection
   b. **HTTP proxy port**
   c. **HTTP proxy login**
   d. **HTTP proxy password**
2. Enter the **Upload address**: only change this address if instructed by technical support.
3. Click **Upload**.

Appendix - Deployment Roles and Related Processes

This section describes Exalead CloudView roles.

Administration Roles

Master

Central administration and management role.

- Required for all deployments
- Unique in the deployment

Related Processes

The `master` process.

Attributes

None.

Gateway

Central administration and management role.

- Hosts many additional administration services
- Required for all deployments
- At most, one per host

Related Processes

The `gateway` process has a single `gatewayPort` attribute.

Attributes

Name	Type	Default	Required	Description
Gateway Port	integer (TCP port)	basePort + 11	Yes	Management API (MAMI) port

Admin UI

This role hosts the Administration Console, the API Console, and the Monitoring Console.

This role is generally unique, and must be coupled with a Gateway role.

Related Processes

Always run within an associated `gateway` process.

Attributes

UI name	XML name	Type	Default	Required	Description
Admin UI port	`adminUIPort`	integer (TCP port)	basePort + 1	Yes	TCP port for the consoles
UI path prefix	`uiPathPrefix`	String	/admin	No	HTTP context path for the Administration Console
API console disabled	`apiConsoleDi`	Boolean	false	No	Disables the API Console
Service console disabled	`serviceConso`	Boolean	false	No	Disables the Service Console
Inspection console disabled	`inspectionCo`	Boolean	false	No	Disables the Inspection Console
Admin console disabled	`adminConsole`	Boolean	false	No	Disables the Administration Console

UI name	XML name	Type	Default	Required	Description
Perf monitoring disabled	`perfMonitori`	Boolean	false	No	Disables the Monitoring Console

Mashup Builder

The Exalead CloudView Mashup Builder interface

- Unique in the deployment
- Coupled with a Gateway role

Related Processes

Always runs within an associated `gateway` process.

Attributes

None.

Business Console

The interface used to perform business tuning such as relevance and alerting.

- Unique in the deployment
- Coupled with a Gateway role

Related Processes

Always runs within an associated `gateway` process.

Attributes

None.

Resource Builder

Internal role used to convert semantic resources. Generally tied to the Gateway role.

Related Processes

None.

Attributes

None.

Data Integration Roles

Connector Server

Server for managed connectors. In the connector configuration, you can associate connectors of the same type (exa, java, dotnet) from various build groups with this connector server.

Related Processes

A `connectors-<instance>` process with the following attributes.

Attributes

UI name	XML name	Type	Default	Required	Description
Type	`type`	One of: exa, java, dotnet	None	Yes	Type of the connector server to deploy
Instance name	`instance`	String	None	Yes	Name of this connector server instance
32-bit	`is32Bits`	Boolean	false	No	Forces the connector server to run as a 32-bit process
Port	`port`	Integer	None	No	Forces the HTTP port for the connector server. This is useful only for fetch on the SharePoint connector.
-	`envOverride:`	String	false	No	Specifies the environment

UI name	XML name	Type	Default	Required	Description
					variable VAR. Replaces any previously defined value for VAR by the value specified here.
-	`envPrepend:V`	String	false	No	Considers the environment variable VAR as a path string, and adds an element as a prefix.
-	`envAppend:VA`	String	false	No	Considers the environment variable VAR as a path string, and adds an element as a suffix.

Crawler Server

Server for the HTTP crawler. In the crawler configuration, you can associate crawlers from different build groups with this crawler server.

Related Processes

A `crawler-<instance>` process.

Attributes

UI name	Name	Type	Default	Required	Description
Instance name	`instance`	String	None	Yes	Name of this crawler server instance

Indexing Roles

Indexing Server

The Indexing Server is the central indexing component for a given build group. You can have one Indexing Server per build group.

Related Processes

An `indexingserver-<buildGroup>` process.

Attributes

UI name	XML name	Type	Default	Required	Description
Build group	`buildGroup`	String	None	Yes	Name of the build group
Papi server port	`papiPort`	integer (TCP port)	basePort + 2 (for bg0)	Yes	Communication port for the HTTP Push API
No. of slices	`nbSlices`	integer	None	Yes	Number of index slices in this build group

Converter

Converts documents to text, HTML, and thumbnails.

Related Processes

A `convert-<instance>` process.

The conversion process. You can define this role one or more times. The `convert-<instance>` process is started.

Attributes

UI name	Name	Type	Default	Required	Description
Instance name	`instance`	String	None	Yes	Name of this crawler server instance

Index

A partial or full replica of a build group's index slices.

Related Processes

Several `index6-*` processes, depending on your configuration.

UI Name	XML Name	Type	Default	Required	Description
Build group	`buildGroup`	String	None	Yes	Name of the build group.
Index slice	`indexSlice`	integer	None	No	If this attribute is left empty, it replicates all slices of the build group. Otherwise, it replicates the slice with this ID only.
Instance name (empty for all)	`instance`	String	None	Yes	Name for this set of slice replicas. Replica names must be unique for a slice.
Use the builder dir	`useSharedDir`	Boolean	false	No	Uses the index builder directory instead of replicating the data. Only possible if

UI Name	XML Name	Type	Default	Required	Description
					this role is on the same host as the corresponding Indexing role.
Runtime config	`runtimeConfi`	String	None	Yes	Name of an index runtime config found in `IndexRuntimes.xml`

Dictionary Builder

Computes the global dictionary. Globally unique and required for all deployments. Generally on the same host as the Gateway, in which case it is embedded in the same process.

Related Processes

None.

Attributes

None.

Search Roles

Search Server

A process that can host Search API and UI services. For high availability and scalability, you can deploy as many Search Servers as required.

Related Processes

A `searchserver-<instance>` process.

Attributes

UI name	XML Name	Type	Default	Required	Description
Instance name	`instance`	String	None	Yes	Name of this search server instance
API port	`apiPort`	Integer (TCP Port)	basePort+10	Yes	TCP port for APIs hosted on this Search Server
UI port	`uiPort`	Integer (TCP Port)	basePort+0	Yes	TCP port for UIs hosted on this Search Server
HTTPS	`uiSSL`	Boolean	false	No	Uses SSL for the hosted Mashup UIs
SSL Certificate	`uiSSLCertifi`	String	None	No	Name of the SSL certificate to use for the Mashup UIs. Use the name listed in the keystore.

Search API

An instance of the Exalead CloudView Search API. You can define multiple Search API configurations. A Search API role defines one instance with one configuration.

Related Processes

Runs within the `searchserver-<searchServerInstance>` process.

Attributes

XML Name	XML Name	Type	Default	Required	Description
Search server instance	`searchServer`	String	None	Yes	Deploys this API to the specified Search Server
Search API config	`apiConfig`	String	None	Yes	Name of the SearchAPI config to deploy

Mashup API

An instance of the Exalead CloudView Mashup API for a single Mashup application.

Related Processes

Runs within the `searchserver-<searchServerInstance>` process.

Attributes

UI name	XML Name	Type	Default	Required	Description
Search server instance	`searchServer`	String	None	Yes	Deploys this API to the specified Search Server
Instance name	`instance`	String	None	Yes	Name of this MashupAPI instance, referenced by MashupUI instances
Mashup application	`applicationI`	String	None	Yes	Name of the Mashup application for this Mashup API

Storage Service

Backend storage for Mashup UI. When using a collaborative widget, you actually enrich your document with data corresponding to the user's inputs. You can store this data using the storage service.

For more information, see "Configuring Data Storage for Collaborative Widgets" in the Exalead CloudView Mashup Builder User's Guide.

Related Processes

Runs within the `searchserver-<searchServerInstance>` process.

Attributes

UI name	XML Name	Type	Default	Required	Description
Instance name	`instance`	String	None	Yes	Name of this Storage Service instance referenced by MashupUI instances
Search server instance	`searchServer`	String	None	Yes	Deploys this Storage Service to the specified Search Server

Mashup UI

An instance of the Exalead CloudView Mashup UI for a single Mashup application.

Related Processes

Runs within the `searchserver-<searchServerInstance>` process.

Attributes

UI name	XML Name	Type	Default	Required	Description
Search server instance	`searchServer`	String	None	Yes	Deploys this UI to the specified Search Server

UI name	XML Name	Type	Default	Required	Description
Instance name	`instance`	String	None	Yes	Name of this Mashup UI instance
HTTP context path	`context`	String	None	Yes	HTTP context path where the Mashup application is deployed; by default, `/mashup-ui`
Storage service instance	`storageservi`	String	None	Yes	Instance name of the Storage Service role
Mashup application	`applicationI`	String	None	Yes	Name of the Mashup application for this Mashup UI. Value: default
Mashup WAR path	`mashupWarPat`	String	None	No	Path to the Mashup WAR file
Mashup description	`mashupWarPat`	String	None	No	Description for the Mashup application

For SSL configuration, see Enable HTTPs for the Mashup UI.

Appendix - CloudView Console Commands

The CloudView console (short: CVConsole) is a command-line interface for CloudView administration and debugging.

When administration is possible using the MAMI, this is the preferred approach. When it is not, you can count on the CVConsole.

cvadmin

apps

generate

- Create an archive file for the application you want to deploy.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`apps-file`	`String`		File path for the application archive file
`include`	`String`		Additional files or folders to include in your archive. Separate using semicolons.
`exclude`	`String`		Files or folders to exclude from your archive. Separate using semicolons.
`debug`	`boolean`	`false`	Set to true to see more log information during the packaging process.
`toDirectory`	`boolean`	`false`	Export the application to a directory instead of an archive file.

- Example: `$DATADIR/bin/cvconsole apps generate debug=false toDirectory=false`

install

- Deploy an application on your instance. The instance must be running.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`apps-file` (required)	`String`		Path to the application you wish to install
`port`	`Port`		Override port number variable.
`instance`	`String`		Override instance name variable
`installdir`	`File`		Override INSTALLDIR variable
`datadir`	`File`		Override DATADIR variable
`hostname`	`String`		Override hostname variable
`noversioncheck`	`boolean`	`false`	Disable version check
`propertyFile`	`File`		File that contains variables mapping
`appIsDirectory`	`boolean`	`false`	If the app is provided as a directory instead of a zip file

- Example: `$DATADIR/bin/cvconsole apps install apps-file=$APPS-FILE noversioncheck=false appIsDirectory=false`

installV6

- Deploy an application on your instance in the DS V6 Installer context. The instance must be running.

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`cvapps-dir` (required)	`File`		Path to the application you wish to install
`app-dir` (required)	`File`		The app installation directory path (directory where the installerV6 installs the application)
`noversioncheck`	`boolean`	`false`	Disable version check
`propertyFile`	`File`		File that contains variables mapping

- Example: `$DATADIR/bin/cvconsole apps installV6 cvapps-dir=$CVAPPS-DIR app-dir=$APP-DIR noversioncheck=false`

box

force-crawl-url

- Synchronously crawl a URL through a crawler.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`crawler` (required)	`String`		Name of the crawler to submit the url to.
`url` (required)	`String`		A url to submit.
`site`	`String`		Consider this url belonging to this site.
`siteRoot`	`boolean`	`false`	Consider this url as a site.
`sitePostRedir`	`String`		Consider this url belonging to this site.

Name	Type	Default value	Description
`priority`	`int`	`3`	Which priority to give to this url. Fastest is 0, slowest is most likely 5 (check crawler configuration).
`group`	`String`		In which group to crawl this url.

- Example: `$DATADIR/bin/cvconsole box force-crawl-url crawler=$CRAWLER url=$URL siteRoot=false priority=3`

submit-urls

- Submit urls to a crawler
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`crawler` (required)	`String`		Name of the crawler to submit the url to.
`url` (required)	`String[]` (multivalued)		A url to submit.
`site`	`String`		Consider this url belonging to this site.
`siteRoot`	`boolean`	`false`	Consider this url as a site.
`sitePostRedir`	`String`		Consider this url belonging to this site.
`priority`	`int`	`3`	Which priority to give to this url. Fastest is 0, slowest is most likely 5 (check crawler configuration).
`group`	`String`		In which group to crawl this url.

Name	Type	Default value	Description
`force`	`boolean`	`false`	Crawl even if the url has already been crawled.

- Example: `$DATADIR/bin/cvconsole box submit-urls crawler=$CRAWLER url= $URL siteRoot=false priority=3 force=false`

connect

upgrade-config

- Upgrade a connector configuration.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`name` (required)	`String`		Name of a Java managed connector

- Example: `$DATADIR/bin/cvconsole connect upgrade-config name=$NAME`

indexing

attach-replica

- Attach a replica to its builder
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup` (required)	`BuildGroup`		Build group on which to act
`slice` (required)	`Slice`		Slice on which to act
`instance` (required)	`SliceInstance`		Instance on which to act

- Example: `$DATADIR/bin/cvconsole indexing attach-replica buildGroup= $BUILDGROUP slice=$SLICE instance=$INSTANCE`

clear-replica

- Clear a replica
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup` (required)	`BuildGroup`		Build group on which to act
`slice` (required)	`Slice`		Slice on which to act
`instance` (required)	`SliceInstance`		Instance on which to act

- Example: `$DATADIR/bin/cvconsole indexing clear-replica buildGroup=$BUILDGROUP slice=$SLICE instance=$INSTANCE`

detach-replica

- Detach a replica from its builder
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup` (required)	`BuildGroup`		Build group on which to act
`slice` (required)	`Slice`		Slice on which to act
`instance` (required)	`SliceInstance`		Instance on which to act

- Example: `$DATADIR/bin/cvconsole indexing detach-replica buildGroup=$BUILDGROUP slice=$SLICE instance=$INSTANCE`

disable-analysis

- Disable the analysis
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing disable-analysis buildGroup=bg0`

disable-import

- Disable the import
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing disable-import buildGroup=bg0`

disable-pushapi

- Disable the PushAPI
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing disable-pushapi buildGroup=bg0`

enable-analysis

- Enable the analysis
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing enable-analysis buildGroup=bg0`

enable-import

- Enable the import
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing enable-import buildGroup=bg0`

enable-pushapi

- Enable the PushAPI
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing enable-pushapi buildGroup=bg0`

indexing-status

- Retrieve the indexing status
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing indexing-status buildGroup=bg0`

licensing

dump-users

- Deprecated: use list-users instead.
- This command is safe to use while CloudView is running.
- No parameter.
- Example: `$DATADIR/bin/cvconsole licensing dump-users`

list-users

- Produce the list of active mashup users (session lasts for 30 days). These are the users that count against the users limit.
- This command is safe to use while CloudView is running.
- No parameter.
- Example: `$DATADIR/bin/cvconsole licensing list-users`

linguistic

compile-fastrules

- Compile a FastRules XML resource file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`input` (required)	`File`		Fast rules definition file
`output` (required)	`File`		Output directory

- Example: `$DATADIR/bin/cvconsole linguistic compile-fastrules tokenizationConfig=tok0 input=$INPUT output=$OUTPUT`

compile-features

- Compile a FeaturesExtractor XML resource file

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`input` (required)	`File`		Features definition file
`output` (required)	`File`		Output directory

- Example: `$DATADIR/bin/cvconsole linguistic compile-features tokenizationConfig=tok0 input=$INPUT output=$OUTPUT`

compile-ontology

- Compile an Ontology XML resource file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`input` (required)	`File`		Ontology definition file
`output` (required)	`File`		Output directory

- Example: `$DATADIR/bin/cvconsole linguistic compile-ontology tokenizationConfig=tok0 input=$INPUT output=$OUTPUT`

compile-semantic-extractor

- Compile a SemanticExtractor XML resource file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`input` (required)	`File`		XML definition file

Name	Type	Default value	Description
`output` (required)	`File`		Output directory

- Example: `$DATADIR/bin/cvconsole linguistic compile-semantic-extractor tokenizationConfig=tok0 input=$INPUT output=$OUTPUT`

compile-synonyms

- Compile a Synonyms XML file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`input` (required)	`File`		Synonyms definition file
`output` (required)	`File`		Output directory

- Example: `$DATADIR/bin/cvconsole linguistic compile-synonyms tokenizationConfig=tok0 input=$INPUT output=$OUTPUT`

convert-ontology-from-skos-to-xml

- Convert a SKOS file to an ontology resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`useAltLabels`	`boolean`	`true`	Use skos:altLabels properties
`useHiddenLabels`	`boolean`	`true`	Use skos:hiddenLabels properties
`hierarchicalEntrie`	`boolean`	`true`	Generate hierarchical display forms
`annotation` (required)	`String`		Root annotation to generate

Name	Type	Default value	Description
`useBroaders`	`boolean`	`true`	Use skos:broader properties
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-ontology-from-skos-to-xml input=$INPUT useAltLabels=true useHiddenLabels=true hierarchicalEntries=true annotation=$ANNOTATION useBroaders=true output=$OUTPUT`

convert-ontology-from-xls-to-xml

- Convert an ontology in Excel format to XML
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-ontology-from-xls-to-xml input=$INPUT output=$OUTPUT`

convert-ontology-from-xml-to-xls

- Convert an ontology XML file to Excel format
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-ontology-from-xml-to-xls input=$INPUT output=$OUTPUT`

convert-sentiment-from-xls-to-xml

- Convert a sentiment Excel format to XML file

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-sentiment-from-xls-to-xml input=$INPUT output=$OUTPUT`

convert-sentiment-from-xml-to-xls

- Convert a sentiment XML file to Excel format
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-sentiment-from-xml-to-xls input=$INPUT output=$OUTPUT`

convert-suggest-from-skos-to-xml

- Convert a SKOS file to a suggest resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`useAltLabels`	`boolean`	`true`	Use skos:altLabels properties
`useHiddenLabels`	`boolean`	`true`	Use skos:hiddenLabels properties
`output` (required)	`File`		Output file

- Example: `$DATADIR/bin/cvconsole linguistic convert-suggest-from-skos-to-xml input=$INPUT useAltLabels=true useHiddenLabels=true output=$OUTPUT`

convert-synonyms-from-skos-to-xml

- Convert a SKOS file to synonyms resources
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`input` (required)	`File`		Input file
`useAltLabels`	`boolean`	`true`	Use skos:altLabels properties
`useHiddenLabels`	`boolean`	`true`	Use skos:hiddenLabels properties
`useNarrowers`	`boolean`	`true`	Use skos:narrower properties
`output_lexical` (required)	`File`		Lexical synonym file (alternative labels)
`output_semantic` (required)	`File`		Semantic synonym file (narrower/broader)
`output_related` (required)	`File`		Related synonym file (related)

- Example: `$DATADIR/bin/cvconsole linguistic convert-synonyms-from-skos-to-xml input=$INPUT useAltLabels=true useHiddenLabels=true useNarrowers=true output_lexical=$OUTPUT_LEXICAL output_semantic=$OUTPUT_SEMANTIC output_related=$OUTPUT_RELATED`

logging

set-logging-level

- Set the logging level
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`level` (required)	`class com.exalead.mercur $LoggingLevel`		Logging level to set
`logger`	`String`		Logger to set
`process` (required)	`Process`		Process name
`install` (required)	`Install`		Install name
`host` (required)	`Host`		Host name

- Example: `$DATADIR/bin/cvconsole logging set-logging-level level=$LEVEL process=$PROCESS install=$INSTALL host=$HOST`

plugins

install

- Install a plugin
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`name`	`String`		Force the plugin name (override what's defined in the plugin)
`file` (required)	`File`		Plugin Zip file

- Example: `$DATADIR/bin/cvconsole plugins install file=$FILE`

list

- List the plugins
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`details`	`boolean`	`false`	Print plugin details

- Example: `$DATADIR/bin/cvconsole plugins list details=false`

remove

- Remove a plugin
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`name` (required)	`String`		Plugin name

- Example: `$DATADIR/bin/cvconsole plugins remove name=$NAME`

update

- Update a plugin
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`name`	`String`		Force the plugin name (override what's defined in the plugin)
`file` (required)	`File`		Plugin Zip file

- Example: `$DATADIR/bin/cvconsole plugins update file=$FILE`

resources

compile

- Call the compile MAMI command
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name

- Example: `$DATADIR/bin/cvconsole resources compile resource=$RESOURCE`

declare-ontology-resource

- Declare a resource for an ontology
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`name` (required)	`String`		Resource name
`group`	`String`	`search-expansion`	Resource group to use

- Example: `$DATADIR/bin/cvconsole resources declare-ontology-resource tokenizationConfig=tok0 name=$NAME group=search-expansion`

declare-synonyms-resource

- Declare a synonyms dynamic resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`tokenizationConfig`	`TokenizationConfig`	`tok0`	Tokenization config to use
`name` (required)	`String`		Resource name
`group`	`String`	`search-expansion`	Resource group to use

- Example: `$DATADIR/bin/cvconsole resources declare-synonyms-resource tokenizationConfig=tok0 name=$NAME group=search-expansion`

delete-version

- Delete a version of a resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name

Name	Type	Default value	Description
`version` (required)	`int`	`0`	The version

- Example: `$DATADIR/bin/cvconsole resources delete-version resource=$RESOURCE version=0`

delete-versions-before

- Delete versions of a resource before version
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name
`before` (required)	`int`	`0`	The version

- Example: `$DATADIR/bin/cvconsole resources delete-versions-before resource=$RESOURCE before=0`

download

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name
`dest` (required)	`File`		The local dest
`format`	`String`		The format if not auto-detected with the filename

- Example: `$DATADIR/bin/cvconsole resources download resource=$RESOURCE dest=$DEST`

get-sample

- Print a sample resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name

- Example: `$DATADIR/bin/cvconsole resources get-sample resource=$RESOURCE`

internal-add-group

- Add an internal group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name
`roles` (required)	`String`		The roles on which the group will be published

- Example: `$DATADIR/bin/cvconsole resources internal-add-group group=$GROUP roles=$ROLES`

internal-add-resource-in-group

- Add an internal resource into an internal group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name
`resource` (required)	`Resource`		The resource name
`type` (required)	`String`		The resource type

- Example: `$DATADIR/bin/cvconsole resources internal-add-resource-in-group group=$GROUP resource=$RESOURCE type=$TYPE`

internal-add-semantic-resource-in-group

- Add an internal resource into an internal group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name
`resource` (required)	`Resource`		The resource name
`type` (required)	`String`		The resource type
`tokenizationConfig` (required)	`TokenizationConfig`		The tokenization config

- Example: `$DATADIR/bin/cvconsole resources internal-add-semantic-resource-in-group group=$GROUP resource=$RESOURCE type=$TYPE tokenizationConfig=$TOKENIZATIONCONFIG`

internal-del-group

- Delete an internal group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name

- Example: `$DATADIR/bin/cvconsole resources internal-del-group group=$GROUP`

internal-del-resource-from-group

- Remove an internal resource from an internal group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name
`resource` (required)	`Resource`		The resource name

- Example: `$DATADIR/bin/cvconsole resources internal-del-resource-from-group group=$GROUP resource=$RESOURCE`

list-groups

- List the configured groups

- This command is safe to use while CloudView is running.
- No parameter.
- Example: `$DATADIR/bin/cvconsole resources list-groups`

list-resources

- List the configured resources
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name

- Example: `$DATADIR/bin/cvconsole resources list-resources group=$GROUP`

list-versions

- List the available version of a resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name

- Example: `$DATADIR/bin/cvconsole resources list-versions resource= $RESOURCE`

prepare-new-version

- Prepare a new version of a raw resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name
`comment`	`String`		An optional comment

- Example: `$DATADIR/bin/cvconsole resources prepare-new-version resource= $RESOURCE`

publish

- Call the publish MAMI command
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`group` (required)	`ResourceGroup`		The group name

- Example: `$DATADIR/bin/cvconsole resources publish group=$GROUP`

rollback-to-version

- Rollback to a version of a resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name
`version` (required)	`int`	`0`	The version
`comment`	`String`		An optional comment

- Example: `$DATADIR/bin/cvconsole resources rollback-to-version resource=$RESOURCE version=0`

upload

- Copy the local source file to the ResourceManager
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`resource` (required)	`Resource`		The resource name
`path` (required)	`File`		The local path
`format`	`String`		The format if not auto-detected with the filename

Name	Type	Default value	Description
`publish`	`boolean`	`false`	True if we want to publish once the file is uploaded
`maxGenKept`	`int`	`0`	Maximum number of generations kept
`comment`	`String`		An optional comment

- Example: `$DATADIR/bin/cvconsole resources upload resource=$RESOURCE path=$PATH publish=false maxGenKept=0`

search

expand-query-to-index-query

- Expands a query
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`query` (required)	`String`		Query to expand

- Example: `$DATADIR/bin/cvconsole search expand-query-to-index-query query=$QUERY`

export-all-query-count

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-all-query-count applicationId=default range=month`

export-long-query-count

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-long-query-count applicationId=default range=month`

export-no-result-query-count

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-no-result-query-count applicationId=default range=month`

export-opened-doc-count

- Export a report result
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
applicationId	String	default	Application Id
range	class com.exalead.search $Range	month	Report range
output	File		Output file

- Example: `$DATADIR/bin/cvconsole search export-opened-doc-count applicationId=default range=month`

export-search-results

- Exports search result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
cache	String	no	Enables query caching. Only valid for streaming=false
debug	String		Logs information according to arguments given
ellql_query	String		ELLQL query to execute
full_hits	int		Number of full hits
output (required)	File		Output file path
query	String		Query to execute
target	String	st0	Search target to use
lang	ISOCode		The global query ISO lang
limits	String		Defines search limits, defined as a KV map
logic	SearchLogic		Search logic to use

Name	Type	Default value	Description
`nhits`	`String`		Sets the number of partial hits to retrieve. Only valid for streaming=false
`of`	`String`	`compactxml`	Output format
`pal`	`boolean`		Sets whether pattern expansions is performed in all languages
`sort`	`String`		Sets sort rules, ex: asc(document_file_size). Only valid for streaming=false
`start`	`int`		First full hit index
`streaming`	`boolean`	`true`	Enables streaming
`timeout`	`String`		timeout=INT: Sets the global timeout. Query timeout will be 75% of the global value. timeout=INT,INT: Sets the query and global timeouts

- Example: `$DATADIR/bin/cvconsole search export-search-results cache=no output=$OUTPUT target=st0 of=compactxml streaming=true`

export-top-long-query

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id

Name	Type	Default value	Description
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-top-long-query applicationId=default range=month`

export-top-no-result-query

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-top-no-result-query applicationId=default range=month`

export-top-opened-doc

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-top-opened-doc applicationId=default range=month`

export-top-query

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-top-query applicationId=default range=month`

export-unique-user-count

- Export a report result
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-unique-user-count applicationId=default range=month`

export-very-long-query-count

- Export a report result
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id
`range`	`class com.exalead.search $Range`	`month`	Report range
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search export-very-long-query-count applicationId=default range=month`

generate-static-application-report

- Generate application related reports
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	ApplicationId

- Example: `$DATADIR/bin/cvconsole search generate-static-application-report applicationId=default`

generate-static-report

- Generate global static report
- This command is safe to use while CloudView is running.
- No parameter.
- Example: `$DATADIR/bin/cvconsole search generate-static-report`

get-static-report

- Get a static report as XML
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`applicationId`	`String`	`default`	Application Id

Name	Type	Default value	Description
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search get-static-report applicationId=default`

get-static-report-status

- Get static reports status
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`output`	`File`		Output file

- Example: `$DATADIR/bin/cvconsole search get-static-report-status`

suggest

compile-suggest-from-file

- Compile a suggest file into a suggest resource
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`file` (required)	`File`		The suggest XML file
`output` (required)	`File`		The directory where the suggest resource will be stored
`tokenizationConfig` (required)	`TokenizationConfig`		The tokenization config

- Example: `$DATADIR/bin/cvconsole suggest compile-suggest-from-file file=$FILE output=$OUTPUT tokenizationConfig=$TOKENIZATIONCONFIG`

create-suggest-file-from-index

- Create a suggest file based on an index field content and a query
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`output` (required)	`File`		The destination file
`searchTarget`	`String`		The search target
`searchLogic`	`SearchLogic`		The search logic
`searchCommand`	`String`		The search command pointing to search logic and search target
`field` (required)	`Index6Field`		The index field
`query`	`String`	`#all`	The query
`subExpr`	`boolean`	`false`	
`subString`	`boolean`	`false`	
`maxEntryLength`	`int`	`50`	
`maxSuggestions`	`int`	`10`	
`sanitizeEntries`	`boolean`	`false`	
`splitSentence`	`boolean`	`false`	
`splitNGrams`	`int`	`0`	
`tokenizationConfig`	`String`	`tok0`	The tokenization config

- Example: `$DATADIR/bin/cvconsole suggest create-suggest-file-from-index output=$OUTPUT field=$FIELD query=#all subExpr=false subString=false maxEntryLength=50 maxSuggestions=10 sanitizeEntries=false splitSentence=false splitNGrams=0 tokenizationConfig=tok0`

dump-suggest-to-xml

- Run a suggest build and dump results to an XML file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`name` (required)	`String`		The Suggest name

Name	Type	Default value	Description
`output` (required)	`File`		Path to the output XML file
`dictionary`	`Dictionary`		Dictionary name for related-terms based suggest

- Example: `$DATADIR/bin/cvconsole suggest dump-suggest-to-xml name=$NAME output=$OUTPUT`

utils

decrypt-password

- Decrypt an encrypted password
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`password` (required)	`String`		The encrypted password

- Example: `$DATADIR/bin/cvconsole utils decrypt-password password=$PASSWORD`

encrypt-password

- Encrypt a password
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`password` (required)	`String`		The password in base64

- Example: `$DATADIR/bin/cvconsole utils encrypt-password password=$PASSWORD`

import-certificate

- Add an existing certificate to the truststore of every product instance. Certificate can either be in DER or PEM format.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`cert-file` (required)	`String`		Public cert file in DER or PEM format.
`alias`	`String`	`jetty`	The alias under which the certificate has to be stored.
`password`	`String`	`exalead`	The keystore password.

- Example: `$DATADIR/bin/cvconsole utils import-certificate cert-file= $CERT-FILE alias=jetty password=exalead`

cvdebug

analysis

analyze

- Analyze a file
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`path`	`File`		File path to send to the analysis
`uri`	`URI`		Uri of document (default: file path)
`config`	`AnalysisConfig`	`default_model`	Analysis config name
`pipeline`	`AnalysisPipeline`	`ap0`	Analysis pipeline name

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	BuildGroup name
`connector`	`Source`	`default`	Name of source

- Example: `$DATADIR/bin/cvconsole analysis analyze config=default_model pipeline=ap0 buildGroup=bg0 connector=default`

box

dump

- Dump URLs and metadata of crawled documents from a crawler's storage (box).
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`crawler` (required)	`String`		Name of the crawler to dump.
`prefix`	`String`		Only dump urls beginning with this prefix.

- Example: `$DATADIR/bin/cvconsole box dump crawler=$CRAWLER`

consolidation

compact-cdih

- Compact the CDIH
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`gen0` (required)	`int`	`0`	The first generation
`gen1` (required)	`int`	`0`	The last generation

- Example: `$DATADIR/bin/cvconsole consolidation compact-cdih gen0=0 gen1=0`

debug-object-graph

- Export a debug view of the object graph
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`outputFile` (required)	`String`		Absolute path to the output file
`workerCount`	`String`	`4`	Number of workers

- Example: `$DATADIR/bin/cvconsole consolidation debug-object-graph outputFile=$OUTPUTFILE workerCount=4`

dump-cdih

- Dump the content of the CDIH to the standard output
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`source`	`Source`		Document source
`prefix`	`String`		Document URI prefix

- Example: `$DATADIR/bin/cvconsole consolidation dump-cdih`

export-document-store

- Export the content of the document storage
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`outputFile` (required)	`String`		Absolute path to the output file

- Example: `$DATADIR/bin/cvconsole consolidation export-document-store outputFile=$OUTPUTFILE`

export-object-graph

- Export the content of the object graph
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`seedNodes`	`String`		URI of the nodes from which to start the export (for partial export only)
`depth`	`int`	`0`	The depth of the export (for partial export only)
`outputFile` (required)	`String`		Absolute path to the output file

- Example: `$DATADIR/bin/cvconsole consolidation export-object-graph depth=0 outputFile=$OUTPUTFILE`

full-compact-cdih

- Full compact the CDIH
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act

- Example: `$DATADIR/bin/cvconsole consolidation full-compact-cdih`

full-compact-storage

- Full compact the storage
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act

- Example: `$DATADIR/bin/cvconsole consolidation full-compact-storage`

get-document-store-stats

- Prints document store stats. A document can hold multiple types. You can have results where you wantedstats on one type only but you have several output.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`type`	`String[]` (multivalued)		Types for which to compute stats. If not provided, stats are computed on all types. type=myType format.
`topLimit`	`int`	`10`	Number of documents to keep in the top

- Example: `$DATADIR/bin/cvconsole consolidation get-document-store-stats topLimit=10`

impact-detection-stats

- Output statistics on the most impacting object types and the top N impacting objects
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`instance`	`ConsolidationServe`		Consolidation server instance on which to act
`source`	`Source`		Document source
`prefix`	`String`		Document URI prefix
`graphMatchingExpre`	`String`		A graph_matching_expressions file.
`top`	`int`	`0`	Additionnaly outputs the top N most impacting objects
`outputFile` (required)	`String`		Path to the output file

- Example: `$DATADIR/bin/cvconsole consolidation impact-detection-stats top=0 outputFile=$OUTPUTFILE`

run-impact-detection

- It simulates the impact for a given set of URI using the registered rules of your consolidation server, but you can also specify a specific set of rules.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`instance` (required)	`ConsolidationServe`		Consolidation server instance on which to act.
`uri` (required)	`String[]` (multivalued)		The URIs on which to run the impact detection. uri=myURI format.
`graphMatchingExpre`	`String`		A graph_matching_expressions file.

- Example: `$DATADIR/bin/cvconsole consolidation run-impact-detection instance=$INSTANCE uri=$URI`

dict

dump-ngrams

- Dump a ngrams dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-ngrams lang=$LANG dictionary=$DICTIONARY`

dump-ngrams-size

- Dump a ngrams dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-ngrams-size lang=$LANG dictionary=$DICTIONARY`

dump-phons

- Dump a phonemes dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-phons lang=$LANG dictionary=$DICTIONARY`

dump-phons-size

- Dump a phonemes dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-phons-size lang=$LANG dictionary=$DICTIONARY`

dump-rt

- Dump a related-terms dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-rt dictionary=$DICTIONARY`

dump-rt-size

- Dump a related-terms dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-rt-size dictionary=$DICTIONARY`

dump-stream

- Dump a dict builder stream
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`path` (required)	`File`		Path to the stream
`lang`	`ISOCode`		The language to consider, all by default
`type`	`DictionaryResource`		The resource type to consider, all by default

- Example: `$DATADIR/bin/cvconsole dict dump-stream path=$PATH`

dump-words

- Dump a words dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-words lang=$LANG dictionary=$DICTIONARY`

dump-words-size

- Dump a words dictionary
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`lang` (required)	`ISOCode`		The language of the dictionary to dump
`dictionary` (required)	`Dictionary`		The dictionary to dump

- Example: `$DATADIR/bin/cvconsole dict dump-words-size lang=$LANG dictionary=$DICTIONARY`

get-resources-size

- Get the size in bytes of a given dictionary resources
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`dictionary` (required)	`Dictionary`		The dictionary to get the resources from
`humanize`	`boolean`	`false`	Humanize the output

- Example: `$DATADIR/bin/cvconsole dict get-resources-size dictionary= $DICTIONARY humanize=false`

index

add-virtual-field

- Add a virtual field
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`name` (required)	`String`		The virtual field name

Name	Type	Default value	Description
`expression` (required)	`String`		The virtual field expression

- Example: `$DATADIR/bin/cvconsole index add-virtual-field buildGroup=bg0 slice=0 name=$NAME expression=$EXPRESSION`

automount

- Mount all index WORM file systems as a regular directories
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`mountPoint` (required)	`String`		Empty directory where to mount the WORM file systems
`flatten`	`boolean`	`false`	Mount as a flat file systems (when using an aggregated WORM file system, hides the generations and serials)
`detach`	`boolean`	`false`	Detach the mounted directory from the console (running the `unmount' command will be required to unmount the WORM file system)

- Example: `$DATADIR/bin/cvconsole index automount mountPoint=$MOUNTPOINT flatten=false detach=false`

cleanup

- Cleanup index directories
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act

- Example: `$DATADIR/bin/cvconsole index cleanup buildGroup=bg0 slice=0`

compact

- Compact the index
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index compact buildGroup=bg0 slice=0 gen0=1 gen1=-1`

convert-date-to-index-time

- Convert a date to index time and Unix timestamp.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`year`	`int`	`0`	year
`month`	`int`	`1`	month (1-12)
`day`	`int`	`1`	day, starting from 1
`hour`	`int`	`0`	hour (0-23)
`min`	`int`	`0`	minute (0-59)

Name	Type	Default value	Description
`sec`	`int`	`0`	second (0-59)

- Example: `$DATADIR/bin/cvconsole index convert-date-to-index-time year=0 month=1 day=1 hour=0 min=0 sec=0`

convert-from-index-time

- Convert an index time to Unix timestamp and date format MM/dd/yyyy hh:mm:ss.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`indextime` (required)	`long`	`0`	The time from the index

- Example: `$DATADIR/bin/cvconsole index convert-from-index-time indextime=0`

convert-timestamp-to-index-time

- Convert an Unix timestamp to index time and date.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`uts` (required)	`long`	`0`	Unix timestamps

- Example: `$DATADIR/bin/cvconsole index convert-timestamp-to-index-time uts=0`

debug-list-content

- Interpret data at the given offset as an inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`offset`	`long`	`0`	The offset at which starts the interpretation
`maxDid`	`int`	`0`	The maximum did of the list

- Example: `$DATADIR/bin/cvconsole index debug-list-content buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 offset=0 maxDid=0`

debug-list-header

- Interpret data at the given offset as an inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`offset`	`long`	`0`	The offset at which starts the interpretation
`maxDid`	`int`	`0`	The maximum did of the list

- Example: `$DATADIR/bin/cvconsole index debug-list-header buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 offset=0 maxDid=0`

dump-alphanum-dict

- Dump an alphanum dictionary
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field`	`Index6Field`	`categories`	The category field
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-dict buildGroup=bg0 slice=0 field=categories gen0=1 gen1=-1`

dump-alphanum-full-pattern-list

- Dump an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`missingCharMarker` (required)	`char`	`0`	The missing char marker

Name	Type	Default value	Description
`wildCardMarker` (required)	`char`	`0`	The wild card marker
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-full-pattern-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 missingCharMarker=0 wildCardMarker=0 word=$WORD kind=$KIND`

dump-alphanum-list

- Dump an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 word=$WORD kind=$KIND`

dump-alphanum-missing-chars-list

- Dump an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`marker` (required)	`char`	`0`	The missing char marker
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-missing-chars-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 marker=0 word=$WORD kind=$KIND`

dump-alphanum-missing-chars-prefix-list

- Dump an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`marker` (required)	`char`	`0`	The missing char marker
`prefix` (required)	`String`		The prefix of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-missing-chars-prefix-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 marker=0 prefix=$PREFIX kind=$KIND`

dump-alphanum-prefix-list

- Dump an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`prefix` (required)	`String`		The prefix of the alphanum of the inverted list

Name	Type	Default value	Description
`kind` (required)	`int`		The kind of the word

- Example: `$DATADIR/bin/cvconsole index dump-alphanum-prefix-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 prefix=$PREFIX kind=$KIND`

dump-attribute-group

- Dump an attribute group using gen0-gen1
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`attrGroup` (required)	`int`	`0`	The attribute group ID
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`attribute`	`Index6Field`		Attribute to dump (default = all)

- Example: `$DATADIR/bin/cvconsole index dump-attribute-group buildGroup=bg0 slice=0 attrGroup=0 gen0=1 gen1=-1`

dump-attribute-group-column-infos

- Dump information on the attributes group columns (ItemStore only)
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`instance`	`SliceInstance`		Instance on which to act
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-attribute-group-column-infos buildGroup=bg0 slice=0 gen0=1 gen1=-1`

dump-attribute-group-headers

- Dump the header of attributes group (ItemStore only)
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-attribute-group-headers buildGroup=bg0 slice=0 gen0=1 gen1=-1`

dump-attribute-group-sizes

- Dump sizes in an attribute group using gen0-gen1
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act

Name	Type	Default value	Description
`attrGroup` (required)	`int`	`0`	The attribute group ID
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`attribute`	`Index6Field`		Attribute the sizes in which to dump (default = all)

- Example: `$DATADIR/bin/cvconsole index dump-attribute-group-sizes buildGroup=bg0 slice=0 attrGroup=0 gen0=1 gen1=-1`

dump-category-list

- Dump a category inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`path` (required)	`String`		The path of the category

- Example: `$DATADIR/bin/cvconsole index dump-category-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 path=$PATH`

dump-cdict

- Dump a category dict using gen0-gen1
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The category or value field
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-cdict buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1`

dump-didlist

- Dump a DidList
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`gen0` (required)	`long`	`0`	The first generation
`gen1` (required)	`long`	`0`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-didlist buildGroup=bg0 slice=0 gen0=0 gen1=0`

dump-field

- Dump a field
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-field buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1`

dump-freelist

- Dump a list of cids that can be recycled during next indexing
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The category or value field
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-freelist buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1`

dump-geofield-wkt

- Dumps geoV2 indexed data into one single WKT file
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`directory` (required)	`String`		Directory where the WKT file will be created
`number`	`int`	`-1`	The number of geometries to dump (-1 means all)

- Example: `$DATADIR/bin/cvconsole index dump-geofield-wkt buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 directory=$DIRECTORY number=-1`

dump-linguistic-streams

- Dump linguistic streams
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`wormfs` (required)	`String`		Path to the WormFS
`path` (required)	`String`		Path inside the WormFS

- Example: `$DATADIR/bin/cvconsole index dump-linguistic-streams wormfs=$WORMFS path=$PATH`

dump-numerical-list

- Dump a numerical inverted list
- This command is safe to use while CloudView is running.

- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`operator` (required)	`String`		The numerical operator to apply
`value` (required)	`double`		The reference value

- Example: `$DATADIR/bin/cvconsole index dump-numerical-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 operator=$OPERATOR value=$VALUE`

dump-numerical-range-list

- Dump a numerical inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`from` (required)	`double`		The lower bound

Name	Type	Default value	Description
`to` (required)	`double`		The upper bound

- Example: `$DATADIR/bin/cvconsole index dump-numerical-range-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 from=$FROM to=$TO`

dump-octrees

- Displays the attributes of a did
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act

- Example: `$DATADIR/bin/cvconsole index dump-octrees buildGroup=bg0 slice=0`

dump-point-distance-list

- Dump a point inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

Name	Type	Default value	Description
`x` (required)	`double`		The reference position's x coordinate
`y` (required)	`double`		The reference position's y coordinate
`distance` (required)	`double`		The reference position's x coordinate

- Example: `$DATADIR/bin/cvconsole index dump-point-distance-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 x=$X y=$Y distance=$DISTANCE`

dump-point-kdtree

- Dumps a point field associated kdtrees to svg files
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`depth`	`int`	`0`	The depth of the inspection
`directory` (required)	`String`		Directory where the SVG files will be created

- Example: `$DATADIR/bin/cvconsole index dump-point-kdtree buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 depth=0 directory=$DIRECTORY`

dump-point-polygon-list

- Dump a point inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`polygon` (required)	`String`		The polygon (format: "x1 y1,x2 y2,x3 y3…")

- Example: `$DATADIR/bin/cvconsole index dump-point-polygon-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 polygon=$POLYGON`

dump-rdict

- Dump a category dict using gen0-gen1
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The category field
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

Name	Type	Default value	Description
`maxCid`	`long`	`-1`	Max cid to dump

- Example: `$DATADIR/bin/cvconsole index dump-rdict buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 maxCid=-1`

dump-rtree-dot

- Dumps RTree as Dot graph
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`directory` (required)	`String`		Directory where the SVG files will be created

- Example: `$DATADIR/bin/cvconsole index dump-rtree-dot buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 directory=$DIRECTORY`

dump-rtree-mbrs

- Dumps RTree MBRs (Minimum Bounding Rectangle)
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index dump-rtree-mbrs buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1`

dump-rtree-svg

- Dumps RTree MBRs (Minimum Bounding Rectangle) as SVG
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`directory` (required)	`String`		Directory where the SVG files will be created
`imageSize`	`int`	`0`	Rescale SVG to the given image size
`withGeoInNodeId`	`int`	`-1`	Also dump geometries contained in the given RTree node Id
`depth`	`int`	`-1`	How many levels of the rtree to dump

Name	Type	Default value	Description
`exactDepthOnly`	`boolean`	`false`	Keep only nodes at given depth (depth must be set)

- Example: `$DATADIR/bin/cvconsole index dump-rtree-svg buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 directory=$DIRECTORY imageSize=0 withGeoInNodeId=-1 depth=-1 exactDepthOnly=false`

dump-vdict

- Dump a value dict using gen0-gen1
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The value field
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`maxCid`	`long`	`-1`	Max cid to dump

- Example: `$DATADIR/bin/cvconsole index dump-vdict buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 maxCid=-1`

estimate-attribute-group-ram-usage

- Estimate the quantity of RAM needed by an attr group
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

Name	Type	Default value	Description
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`attrGroup` (required)	`int`	`0`	The attribute group ID

- Example: `$DATADIR/bin/cvconsole index estimate-attribute-group-ram-usage buildGroup=bg0 slice=0 attrGroup=0`

field-dict-stats

- Display the statistics of the dictionary of a given field
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

- Example: `$DATADIR/bin/cvconsole index field-dict-stats buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1`

full-compact

- Full compact the index
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`ignoreGlobalSuppor`	`boolean`	`false`	Ignore global support (advanced setting, used during migration to 16x)

- Example: `$DATADIR/bin/cvconsole index full-compact buildGroup=bg0 slice=0 ignoreGlobalSupport=false`

get-attributes

- Displays the attributes of a did
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`did` (required)	`int`		The did

- Example: `$DATADIR/bin/cvconsole index get-attributes buildGroup=bg0 slice=0 did=$DID`

get-did-states

- Display the states of a did in all slots
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`did` (required)	`int`		The did

- Example: `$DATADIR/bin/cvconsole index get-did-states buildGroup=bg0 slice=0 did=$DID`

inspect-alphanum-full-pattern-list

- Inspect an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`missingCharMarker` (required)	`char`	`0`	The missing char marker
`wildCardMarker` (required)	`char`	`0`	The wild card marker
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-alphanum-full-pattern-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 missingCharMarker=0 wildCardMarker=0 word=$WORD kind=$KIND depth=0`

inspect-alphanum-list

- Inspect an alphanum inverted list

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-alphanum-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 word=$WORD kind= $KIND depth=0`

inspect-alphanum-missing-chars-list

- Inspect an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name

Name	Type	Default value	Description
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`marker` (required)	`char`	`0`	The missing char marker
`word` (required)	`String`		The word of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-alphanum-missing-chars-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 marker=0 word=$WORD kind=$KIND depth=0`

inspect-alphanum-missing-chars-prefix-list

- Inspect an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`marker` (required)	`char`	`0`	The missing char marker

Name	Type	Default value	Description
`prefix` (required)	`String`		The prefix of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-alphanum-missing-chars-prefix-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 marker=0 prefix=$PREFIX kind=$KIND depth=0`

inspect-alphanum-prefix-list

- Inspect an alphanum inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`prefix` (required)	`String`		The prefix of the alphanum of the inverted list
`kind` (required)	`int`		The kind of the word
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-alphanum-prefix-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 prefix=$PREFIX kind=$KIND depth=0`

inspect-category-list

- Inspect a category inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`path` (required)	`String`		The path of the category
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-category-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 path=$PATH depth=0`

inspect-didlist

- Inspect a did list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act

Name	Type	Default value	Description
`instance`	`SliceInstance`		Instance on which to act
`gen0` (required)	`long`	`0`	The first generation
`gen1` (required)	`long`	`0`	The last generation
`depth`	`int`	`0`	The depth of inspection

- Example: `$DATADIR/bin/cvconsole index inspect-didlist buildGroup=bg0 slice=0 gen0=0 gen1=0 depth=0`

inspect-didlists

- Inspect a set of did lists
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`depth`	`int`	`0`	The depth of inspection

- Example: `$DATADIR/bin/cvconsole index inspect-didlists buildGroup=bg0 slice=0 gen0=1 gen1=-1 depth=0`

inspect-field

- Inspect a field
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

Name	Type	Default value	Description
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-field buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 depth=0`

inspect-numerical-list

- Inspect a numerical inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`operator` (required)	`String`		The numerical operator to apply
`value` (required)	`double`		The reference value
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-numerical-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 operator=$OPERATOR value=$VALUE depth=0`

inspect-numerical-range-list

- Inspect a numerical inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`from` (required)	`double`		The lower bound
`to` (required)	`double`		The upper bound
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-numerical-range-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 from=$FROM to=$TO depth=0`

inspect-point-distance-list

- Inspect a point inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act

Name	Type	Default value	Description
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`x` (required)	`double`		The reference position's x coordinate
`y` (required)	`double`		The reference position's y coordinate
`distance` (required)	`double`		The reference position's x coordinate
`depth`	`int`	`0`	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-point-distance-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 x=$X y=$Y distance=$DISTANCE depth=0`

inspect-point-kdtree

- Inspect a point field associated kdtrees
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation

Name	Type	Default value	Description
`depth`	`int`	0	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-point-kdtree buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 depth=0`

inspect-point-polygon-list

- Inspect a point inverted list
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	bg0	Build group on which to act
`slice`	`Slice`	0	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`field` (required)	`Index6Field`		The field name
`gen0`	`long`	1	The first generation
`gen1`	`long`	-1	The last generation
`polygon` (required)	`String`		The polygon (format: "x1 y1,x2 y2,x3 y3…")
`depth`	`int`	0	The depth of the inspection

- Example: `$DATADIR/bin/cvconsole index inspect-point-polygon-list buildGroup=bg0 slice=0 field=$FIELD gen0=1 gen1=-1 polygon=$POLYGON depth=0`

inspect-support

- Inspect the support
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`gen0`	`long`	`1`	The first generation
`gen1`	`long`	`-1`	The last generation
`depth`	`int`	`0`	The depth of inspection

- Example: `$DATADIR/bin/cvconsole index inspect-support buildGroup=bg0 slice=0 gen0=1 gen1=-1 depth=0`

mount

- Mount a WORM file system as a regular directory
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`mountPoint` (required)	`String`		Empty directory where to mount the WORM file system
`flatten`	`boolean`	`false`	Mount as a flat file system (when using an aggregated WORM file system, hides the generations and serials)
`detach`	`boolean`	`false`	Detach the mounted directory from the console (running the `unmount' command will

Name	Type	Default value	Description
			be required to unmount the WORM file system)

- Example: `$DATADIR/bin/cvconsole index mount buildGroup=bg0 slice=0 mountPoint=$MOUNTPOINT flatten=false detach=false`

query

- Perform a query written in ELLQL
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act
`ellql` (required)	`String`		The ELLQL query
`virtualFields`	`String[]` (multivalued)		Sets virtual fields name=expr format
`verbose`	`boolean`	`false`	Displays all the DID

- Example: `$DATADIR/bin/cvconsole index query buildGroup=bg0 slice=0 ellql=$ELLQL verbose=false`

remove-virtual-field

- Remove a virtual field
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`slice`	`Slice`	`0`	Slice on which to act
`instance`	`SliceInstance`		Instance on which to act

Name	Type	Default value	Description
`name` (required)	`String`		The virtual field name

- Example: `$DATADIR/bin/cvconsole index remove-virtual-field buildGroup=bg0 slice=0 name=$NAME`

unmount

- Unmount a WORM file system detached from the console
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`mountPoint` (required)	`String`		Directory where a WORM file system detached from the console has been mounted

- Example: `$DATADIR/bin/cvconsole index unmount mountPoint=$MOUNTPOINT`

indexing

compact-dih

- Compact the DIH
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`gen0` (required)	`int`	`0`	The first generation
`gen1` (required)	`int`	`0`	The last generation

- Example: `$DATADIR/bin/cvconsole indexing compact-dih gen0=0 gen1=0`

compact-document-cache

- Compact the document cache (must be done offline)

- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`gen0` (required)	`int`	`0`	The first generation
`gen1` (required)	`int`	`0`	The last generation

- Example: `$DATADIR/bin/cvconsole indexing compact-document-cache gen0=0 gen1=0`

dump-checkpoint-store

- Dump the content of the checkpoint store to the standard output
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`source`	`Source`		Checkpoint source

- Example: `$DATADIR/bin/cvconsole indexing dump-checkpoint-store`

dump-dih

- Dump the content of the DIH to the standard output
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`source`	`Source`		Document source
`prefix`	`String`		Document URI prefix

- Example: `$DATADIR/bin/cvconsole indexing dump-dih`

dump-document-cache

- Dump the content of the document cache to the standard output
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`source`	`Source`		Document source
`prefix`	`String`		Document URI prefix
`dumpDirectives`	`boolean`	`true`	Dump document and parts directives
`dumpMetas`	`boolean`	`true`	Dump document metas
`dumpParts`	`boolean`	`true`	Dump document parts

- Example: `$DATADIR/bin/cvconsole indexing dump-document-cache dumpDirectives=true dumpMetas=true dumpParts=true`

export-document-cache

- Export the content of the document cache on the filesystem
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act
`source`	`Source`		Document source
`prefix`	`String`		Document URI prefix
`exportDirectives`	`boolean`	`true`	Export document and parts directives
`exportMetas`	`boolean`	`true`	Export document metas
`exportParts`	`boolean`	`true`	Export document parts

Name	Type	Default value	Description
`outputDir` (required)	`String`		Output directory

- Example: `$DATADIR/bin/cvconsole indexing export-document-cache exportDirectives=true exportMetas=true exportParts=true outputDir= $OUTPUTDIR`

full-compact-dih

- Full compact the DIH
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing full-compact-dih`

full-compact-document-cache

- Full compact the document cache
- This command is unsafe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`		Build group on which to act

- Example: `$DATADIR/bin/cvconsole indexing full-compact-document-cache`

linguistic

dump-ontology

- Dump an ontology
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`path` (required)	`File`		Path to compiled ontology

- Example: `$DATADIR/bin/cvconsole linguistic dump-ontology path=$PATH`

process

create-launcher

- Create a script to launch a process with various arguments (Linux ONLY)
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`process` (required)	`Process`		Name of the process

- Example: `$DATADIR/bin/cvconsole process create-launcher process= $PROCESS`

run

- Start a process
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`process` (required)	`Process`		Name of the process to start
`debugmalloc`	`boolean`	`false`	Enable debugmalloc
`debugger`	`boolean`	`false`	Run with debugger
`attachdebugger`	`boolean`	`false`	Attach a debugger in case of crash
`env`	`String[]` (multivalued)		Additional environment

- Example: `$DATADIR/bin/cvconsole process run process=$PROCESS debugmalloc=false debugger=false attachdebugger=false`

show-command

- Show the command used to start a process
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`process` (required)	`Process`		Name of the process to start

- Example: `$DATADIR/bin/cvconsole process show-command process=$PROCESS`

regex

match

- Parse, compile the specified regular expression, then eventually match an input string and substitute the matches in a sed like output format.
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`regex` (required)	`String`		expression to be compiled. Supports most Perl 5 syntax
`input`	`String`		string to be matched by the regex
`replacement`	`String`		string used to replace the match in the input string. Supports sed-like output format
`caseInsensitive`	`boolean`	`false`	match string with no distinction between capital letters and minuscules

Name	Type	Default value	Description
`normalized`	`boolean`	`false`	match string regardless the case and accent differences
`leftAnchor`	`boolean`	`false`	match only at the beginning of the input
`rightAnchor`	`boolean`	`false`	match only at the end of the input

- Example: `$DATADIR/bin/cvconsole regex match regex=$REGEX caseInsensitive=false normalized=false leftAnchor=false rightAnchor=false`

semantic

annotate

- Process a string and dump the generated annotated tokens
- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`value`	`String`		Text to process, standard input if missing
`language`	`ISOCode`		Iso code of the language
`context`	`String`		Context of the chunk
`pipeline`	`AnalysisPipeline`	`ap0`	Analysis pipeline to use

- Example: `$DATADIR/bin/cvconsole semantic annotate buildGroup=bg0 pipeline=ap0`

annotate-file

- Process an entire text file and dump the generated annotated tokens

- This command is safe to use while CloudView is running.
- Parameters:

Name	Type	Default value	Description
`buildGroup`	`BuildGroup`	`bg0`	Build group on which to act
`path` (required)	`File`		Path to file to process
`encoding`	`String`	`UTF-8`	File encoding
`language`	`ISOCode`		Iso code of the language
`context`	`String`		Context of the chunk
`pipeline`	`AnalysisPipeline`	`ap0`	Analysis pipeline to use

- Example: `$DATADIR/bin/cvconsole semantic annotate-file buildGroup=bg0 path=$PATH encoding=UTF-8 pipeline=ap0`

dump-pipe

- Dump the current MOT pipe
- This command is safe to use while CloudView is running.
- No parameter.
- Example: `$DATADIR/bin/cvconsole semantic dump-pipe`

Appendix - CloudView Support Information

This appendix describes several concepts and best practices that are useful for Exalead CloudView Support activities. These are basics that you must know before contacting the Exalead Support or R&D team.

CloudView Processes

To monitor your application more efficiently, it is important to understand how indexing and search work and which process intervenes at each step of the Exalead CloudView workflow.

For more information about the processes involved at each step of the Exalead CloudView workflow, see Processes in Detail.

To monitor processes and check performance issues, see Technical Monitoring.

Key Configuration Concepts (Edit, Apply, Rollback)

Exalead CloudView has a versioned configuration.

When you save a modification in the Administration Console, the new configuration does not immediately apply to the running instance. Instead, it appears in the configuration store.

When you apply the configuration, it shifts to all running components. Exalead CloudView checks the high-level configuration for errors, then computes a low-level configuration for each process, known as the GCT.

You can find the configuration store in `<DATADIR>/config`. It contains the latest version of all configurations, stored as XML files.

To roll back to a previous configuration, you must:

1. From the top navigation bar of the Administration Console, click the down arrow next to **Apply**.
2. Select **Rollback to version** for the required version.

For more information, see Managing Configurations.

MAMI Basics (API-UI, cvcommand)

The Management API (MAMI) is the public API that allows you to administer and configure Exalead CloudView.

You can get a raw access to this API at the following URL: `http://<HOSTNAME>:<BASEPORT+11>`.

Note: The Administration Console and the API Console are 2 console UIs, designed to configure the MAMI and the Push API (AKA PAPI for the connector configuration) underneath.

For more information, see in the Exalead CloudView Programmer's Guide.

cvcommand Command-Line Tool

You can use the `<DATADIR>\bin\cvcommand` command-line tool to run several administration commands to any running Exalead CloudView product. For example, back up and restore operations.

For more information, see cvcommand and cvcmd and Backup/Restore Operations.

cvdebug Command-Line Tool

You can use the `<DATADIR>\bin\cvdebug` command-line tool to perform advanced analysis and indexing debug.

For more information, see cvdebug.

Remove Useless Files

You can remove useless files to reduce disk space. For more information, see Controlling Disk Space.

Generate a cvdiag

Exalead CloudView includes its own system reporting mechanism. When an error occurs, the Exalead Support team asks you to send them a CVDiag (abbreviation for Exalead CloudView Diagnostics).

To send a CVDiag:

1. In the Administration Console, go to **Help > Create system report**.
2. Click **Download**.

A `cvdiag<timestamp>.zip` file is downloaded. For example,
`cvdiag_20190829-102718.653.zip`.

3. Unzip the file.
4. Identify the exact Exalead CloudView version that is running with one of the following methods:

- In the Administration Console, go to **Help > About Cloudview**.
- In the unzipped CVDiag, open `kit_files/productversion.txt`.

5. In the unzipped CVDiag, identify the applied Hotfix level, using the `hotfixes/cumulative/manifest.xml` file.

   Recommendation: Always verify the hotfix level and install the latest hotfix before contacting the Exalead R&D team.

For more information, see Sending System Reporting.

Open the cvdiag perfmonitoring on Your Machine

You can use the Exalead CloudViewMonitoring Console to monitor performance.

For example, this is how to see a file system full after an index compact operation.

1. Stop Exalead CloudView on your Support instance: `<DATADIR>\bin\cvinit.sh|bat stop`.
2. In your `DATADIR`, rename your `perfmonitoring` subfolder to avoid overwriting it with the customer's one. For example, `perfmonitoring`.
3. In your cvdiag folder, copy the `perfmonitoring` subfolder, for example, `cvdiag_20190829-102718.653\perfmonitoring`.
4. Paste the cvdiag `perfmonitoring` subfolder in your `DATADIR`.
5. Restart Exalead CloudView.
6. Open the Monitoring Console: `http://<HOSTNAME>:<BASEPORT+1>/perf-ui`.

For more information about `perfmonitoring` analysis, see Check system health and services performance.

Replay a Query from search.csv

Exalead CloudView contains several reporters allowing you to collect reporting information related to the behavior of your front-end applications.

By default, the search-reporting reporter is active and collects user query data in search.csv file.

1. Unzip your cvdiag.
2. Go to `<CVdiag dir>/run/<instance name>-cvdefault/searchserver-ss0/search-reporting`.
3. Edit the `search.csv` file.

The default configuration includes many fields revealing information such as the query made, the number of hits returned, total query time, etc.

For more information, see in the Exalead CloudView Configuration Guide.

Get Java/Native Stacks

For each Exalead CloudView process, you can get Java and Native stacks from the MAMI at: `http://<HOSTNAME>:<BASEPORT+11>/services`.

Verify Applied Hotfixes

About FDs and Hot Fixes

During a release lifecycle of the 3D#EXPERIENCE platform, Dassault Systèmes provides Fix Deliveries (FD), also called Fix Packs (FP), to fix issues or deliver new and enhanced features. Each FD corresponds to a technical release of Exalead CloudView that packages components called hot fixes (HF).

Updating the 3D#EXPERIENCE platform with FDs, involves deploying the same FD version of all components already installed on the 3D#EXPERIENCE platform. This is to guarantee that they are compliant with each other but this operation is heavy and may take a lot of time.

For Exalead CloudView, to correct issues more quickly, you can deploy the latest hotfix of the technical release only. These hotfixes are cumulative so you can install the latest without installing all intermediary versions.

Do You Use the Latest Hot Fix?

The first thing you must do is to verify whether your installation uses the latest cumulative hotfix. To do so:

1. In the Administration Console, go to **Help > About > Hotfixes** and look at the hotfix applied on your installation.
2. Generate a CVDiag (see Generate a cvdiag), unzip it, and identify the applied Hotfix level, using the `hotfixes/cumulative/manifest.xml` file.
3. On Airbridge, go to **CloudView > Downloads > All releases**, select your release and verify the latest hotfix version using the `README.TXT` file.
4. If a new hotfix is available, download it from Airbridge and install it as described in the following section.

Install Hot Fixes

Important: Install the Hot Fix media in the same directory as the GA version. An Hot Fix installation is always performed in "delta" mode. This means that the Hot Fix media does not contain all files: it contains only the files that are different from the previous Hot Fix or from the GA level of the same version.

For more information on how to install Hotfixes, see in the Exalead CloudView Installation Guide.

Search API Basics

The Search API is the entry point for performing searches on Exalead CloudView. It provides a public HTTP interface to access the commands defined in the Exalead CloudView Search API configuration.

By default, when you install an Exalead CloudView product, the commands are available at `http://<HOSTNAME>:<BASEPORT+10>/<COMMAND_PATH>`.

For example, a Search command called "`search-api`" is installed, and is available at `http://<HOSTNAME>:<BASEPORT+10>/search-api`.

For more information about the Search API concepts, see in the Exalead CloudView Programmer's Guide.

For a complete list of parameters that you can use in this API, see in the Exalead CloudView Configuration Guide.

What is an ELLQL Query

Exalead CloudView Low-Level Query Language (ELLQL) is mostly used for programmatic generation of queries, similar to SQL. ELLQL is typically used by custom programs to enrich user queries and to add additional features. Internally, all user-entered UQL queries are transformed into ELLQL.

Note: You can pass ELLQL queries via the Search API using the eq parameter instead of q for UQL queries.

For more information, see in the Exalead CloudView Configuration Guide.

Difference Between mashup-ui and search-api

You can use either the Mashup API or the Search API for search in your applications.

For more information, see in the Exalead CloudView Mashup Programmer's Guide.

What is a Security Source and Security Tokens

Document security is implemented by indexing a document’s Access Control List (ACL) and generating security tokens when the user authenticates.

In Exalead CloudView, you must configure a security source to authenticate users and authorize their document access.

For a simple introduction, see in the Exalead CloudView Getting Started Guide.

For more details, see Managing User Access.

Where Can I Find Log Files in the cvdiag and the datadir

Logs are available for each process. They are written to `<DATADIR>\run\<PROCESS NAME>\log.log` and can be displayed from **Administration Console > Logs**.

You can find a global log file gathering all process logs in `<DATADIR>\run\global.log`.

To reduce disk space used by logs, you can:

- Change log location
- Rotate and purge logs

For more information, see Configuring Logs.

Note: To retrieve log files from a cvdiag, go to `<CVdiag dir>/run/<instance name>-cvdefault/<process name>/log.log`.

For example, `cvdiag_20190829-102718.653\run\myexampleserver-cvdefault\indexingserver-bg0\log.log`.

Configure Logs in Debug Mode

In the Administration Console, logs are configured to **Info** logging level by default.

To debug your Exalead CloudView configuration, configure the **Default logging level** property to **Debug**.