

CloudView CV23

# Getting Started

# Table of Contents

Getting Started.....	4
What's New.....	5
What is Exalead CloudView.....	6
What Are the Main Parts of Exalead CloudView.....	6
Exalead CloudView Software Interfaces.....	7
Exalead CloudView Terminology.....	8
Creating an Application (Methodology).....	10
Installing Exalead CloudView on Windows.....	13
Install Exalead CloudView.....	13
Check the Installation Logs If Errors Occur.....	14
Configure Exalead CloudView Using the Setup Wizard.....	14
Access the Configuration Interfaces.....	20
Creating a Simple Search Application.....	22
Defining the Application Scope.....	22
Tutorial: What Is Our Data Source.....	23
Tutorial: What We Want to Do Functionally.....	23
Adding a Connector.....	23
Creating the Data Model.....	25
The Data Model in a Nutshell.....	25
Configuring Data Model Properties.....	26
Create a New Class in the Data Model.....	28
Configuring the Search Bar Behavior.....	33
Use Full-Text Search.....	33
Use Prefix Search to Make Specific Queries.....	34
Use Semantic Analysis to Correct Queries.....	36
Preparing the Content to Display in the Search Results.....	39
Select the Hit Metas and Hit Facets to Display.....	40
Highlight Query Terms in Search Results.....	42
Summarize the Content of Hit Metas.....	43
Defining the Refinement Facets.....	45
Tutorial: Enhance the Refinement Facets Display.....	45
Tutorial: Add a Numerical Range facet for Unit Prices.....	45
Creating the Front-end Application with Mashup Builder.....	48
About Mashup Builder.....	49
Modify the Display of Hit Titles.....	49
Change the Navigation Facets Order.....	50
Going Further with CloudView.....	54
Tools for Analytics.....	54
Tutorial: Aggregate Facet Values.....	55
Tutorial: Create a Multi-Dimension Facet.....	57
Tutorial: Add a Pivot Table.....	59
Tutorial: Enable Drill-Down Using Hierarchical Facets.....	61
Collapse or Group Search Results.....	64
Expanding User Queries.....	65
About Query Expansion Features.....	65
Tutorial: Add Suggest on the Name Field.....	67
Related Terms.....	68
Consolidating Data.....	68
Configuring Results Ordering and Grouping.....	69

Creating Query Alerts..... 69

Securing Document Access.....69

    How Document Security Works.....70

    Tutorial: Implement Document Security.....72

Customizing CloudView..... 74

# Getting Started

This guide provides step-by-step instructions for creating a simple search application, explaining core Exalead CloudView concepts along the way. You will install Exalead CloudView, then work in the Administration Console and the Mashup Builder.

## Audience

This guide is mainly destined to consultants, developers, or system administrators who are new to Exalead CloudView.

## Further Reading

You might need to refer to the following guides:

Guide	for more details on
Installation & Administration	installing the product and administrative tasks typically performed on production systems.
Connectors	standard connectors' configuration.
Configuration	indexing and search concepts, as well as advanced functionality.
Mashup Builder	building the front end of your search application.

# What's New

There are no enhancements in this release.

# What is Exalead CloudView

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

The Exalead CloudView platform uses textual and semantic technologies to reconcile formats, structures and terminologies, and identify embedded meanings and relationships. All information sources are unified in a robust modular index that ensures continuous access and optimal use of server resources.

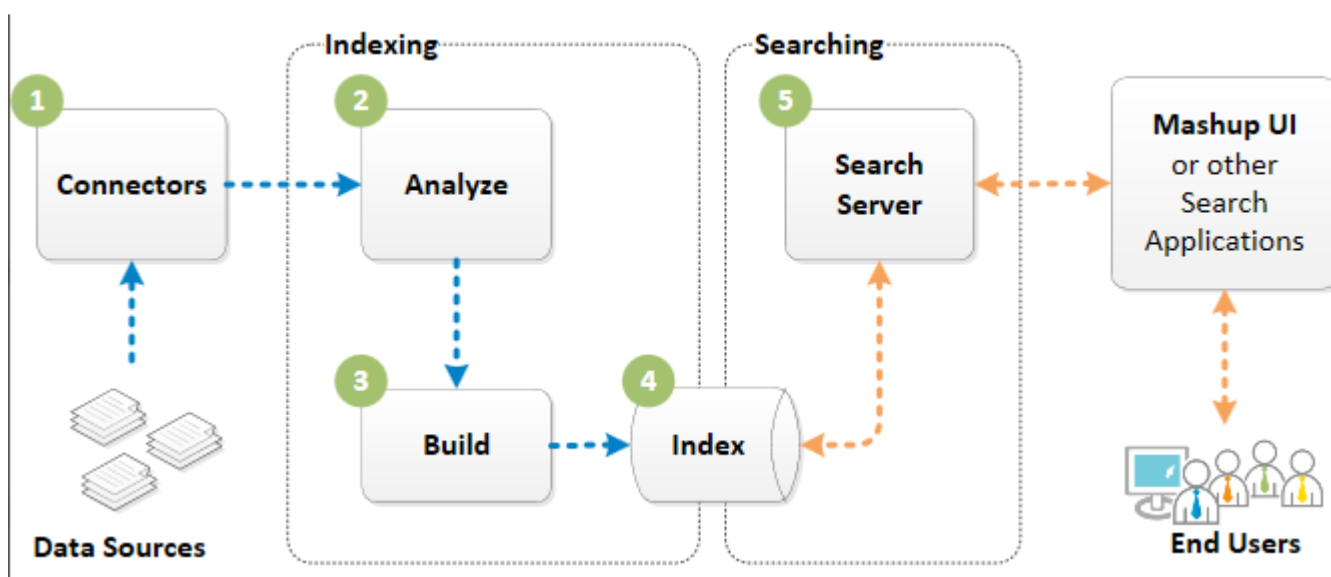
## What Are the Main Parts of Exalead CloudView

To start with a high-level view of the product, let us say that it is made of the following parts:

- Connectors to fetch data from data sources.
- Indexing to process fetched documents and store them into the index.
- Searching to define how data will be searchable and what will be displayed in the Search-Based Application.
- Mashup UI or custom front-end applications based on search.

The following diagram summarizes the process to index and search with Exalead CloudView.

*A simplified view of Exalead CloudView*



#	Description
---	-------------

- |   |  |
|---|--|
| 1 | Connectors: <ul style="list-style-type: none"> <li>• access the data sources,</li> </ul> |
|---|--|

#	Description
	<ul style="list-style-type: none"> <li>• fetch their files,</li> <li>• convert the files into documents,</li> <li>• send them to the Indexing Server through the Push API protocol.</li> </ul>
2	<p>During the analysis phase, the Indexing Server:</p> <ul style="list-style-type: none"> <li>• Receives documents.</li> <li>• Triggers their analysis sequentially, entirely in memory.</li> <li>• The analyzers process each document in the analysis job, perform text extraction, semantic processing, custom operations, and mapping.</li> </ul>
3	<p>During the build phase, the Indexing Server analyzes pushed documents, and creates a new generation of the index. It creates a set of files (tables, inverted list, and other structures) to make the index efficient at search time.</p>
4	<p>Once the build phase is complete, it is stored into the index.</p> <ul style="list-style-type: none"> <li>• It merges the data computed for analysis with the current version of the index.</li> <li>• Once done, the index is committed and updated. The new documents are available for search.</li> </ul>
5	<p>The Search Server interprets and processes user queries.</p> <p>Each user query is processed by the Search Server based on a specific search logic.</p>
6	<p>Search queries and search results can be entered and displayed either in the Mashup UI (the default search application), or a custom search application relying on the Exalead CloudView Search API.</p>

## Exalead CloudView Software Interfaces

### Main Software Interfaces

Exalead CloudView includes the following main software interfaces:

- Administration Console
- Mashup UI
- Mashup Builder
- Business Console

- Monitoring Console
- API Console

For more details on how to display these software interfaces, see [Access the Configuration Interfaces](#).

## Software Extensions

The following software interface extensions are available as options. Contact your Exalead representative for your licensing needs.

- Mashup Builder Premium
- Content Recommender extends the Business Console with a business logic recommendation engine to recommend results depending on a search query, for example, to recommend Home-cinemas when the user searches for TV sets.

## Add-ons

Add-ons are optional components extending the capabilities of Exalead CloudView. You must install them in the `INSTALLDIR`. They require additional installation steps and a product restart to be functional. Exalead CloudView add-ons include:

- Extended Languages add-on extends semantic analysis for a wider variety of languages.
- Russian Lemmatization add-on allows Exalead CloudView to lemmatize Russian. As this language contains many inflections and therefore involves a large resource, it is not shipped by default.

## Exalead CloudView Terminology

- Connectors provide access to your data source (files, records), converts them into Exalead CloudView documents, and then sends them to Exalead CloudView for indexing. Connectors use the Push API (PAPI), a simple HTTP API to feed the index with documents. Each connector relies on the data source's native protocol to connect to its information source.
- Convert allows Exalead CloudView to read documents with various formats (such as PDF, XML, or Microsoft Word). It receives documents from connectors, extract text and field information from them, and pass that information along for indexing and storage in the index.
- Corpus refers to the collection of documents, coming from one or several data sources that needs to be indexed.
- Documents can be defined as all the objects to be indexed by Exalead CloudView, regardless of file or entity type in the data source. For example, HTML, JPG or CSV files, database



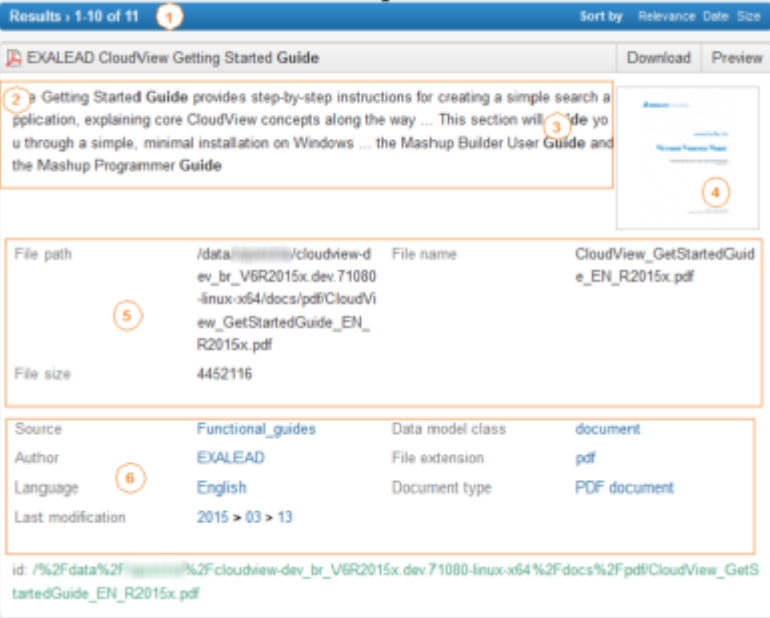
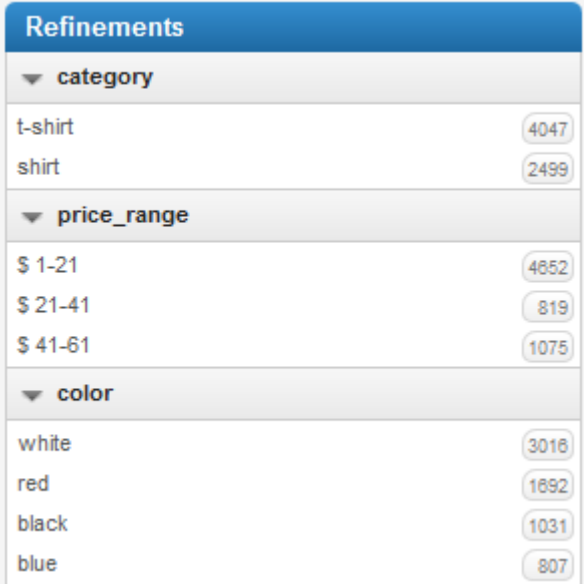
records are all considered documents within Exalead CloudView, since they are all converted into a Exalead CloudView-specific document format (also known as a PAPI document) after being scanned by a connector.

- Document metas, not to be confused with hit metas, are pieces of text belonging to a document that have associated values, such as title or size. Document metas are stored either as an index field or as a category. Context is sometimes used as a synonym for document meta.
- Dictionary is a separate structure from the index that stores all the words from an indexed document, plus their number of occurrences in the corpus. It is used for linguistic expansion mechanisms such as spell-checking or regular expression matching.
- Facets are used to narrow search results. Use them to drill down into an area, such as language, author, or file type. They are typically used in dashboard analytics widgets, or in the Refinements panel for enterprise search.
- Hit metas, not to be confused with document metas, are used to display one or more retrievable index fields in the hit content of search results.
- Index is an efficient structure used by Exalead CloudView to store information about the items it has analyzed. When users issue search queries, Exalead CloudView quickly and easily finds the results within this structure.
- The Exalead CloudView index is divided into fields:
  - Each field has a type: alphanumeric, numeric, hierarchical categories, geographic, and so forth.
  - Each field can be defined as:
    - Searchable which means that user search queries can be applied to this field.
    - Retrievable which means that the field can be displayed in the search results.
- Queries are the search requests sent to the Exalead CloudView search engine and processed according to a specified search logic.
- Thumbnails are small image previews for documents, which can be displayed in the search results. They are computed at search-time and kept in the browser cache for one week.

# Creating an Application (Methodology)

This section describes the main steps to create a Search-Based Application. Read it carefully and click the links to get to the procedures detailing the configuration of each step.

Step	Description	See ...
1	Define the Search-Based Application scope. <ul style="list-style-type: none"> <li>Which data sources do you have?</li> <li>What do you want to search (data to retrieve)?</li> <li>What do you want to show in your Search-Based Application?</li> </ul>	<a href="#">Defining the Application Scope</a>
2	Create Connectors. <ul style="list-style-type: none"> <li>Connectors fetch documents from data sources and push them into Exalead CloudView.</li> <li>Exalead CloudView converts them into Exalead CloudView documents.</li> </ul>	<a href="#">Adding a Connector</a>
3	Create the Data Model. Create the metas and facets that you want to use in your application (see <a href="#">Indexing Options for Data Model Properties</a> ): <ul style="list-style-type: none"> <li>Retrievable index fields - the fields to display in the search results.</li> <li>Searchable index fields - the fields in which you can search data.</li> <li>Facetable fields - category facets to display in the Refinement panel to refine and navigate through search results.</li> </ul>	<a href="#">Creating the Data Model</a>
4	Configure the behavior of the search fields. <ul style="list-style-type: none"> <li>Full-text search</li> <li>Prefix search</li> <li>Semantic processing</li> </ul>	<a href="#">Configuring the Search Bar Behavior</a>
5	Prepare the content to display in search results (hits). <ul style="list-style-type: none"> <li>(1) Number of hits to display in the page</li> <li>(2) Summary descriptions</li> <li>(3) Highlighting of terms</li> </ul>	<a href="#">Preparing the Content to Display in the Search Results</a>

Step	Description	See ...
	<p>(4) Browsable thumbnails</p> <p>(5) Hit metas</p> <p>(6) Hit facets</p> 	
6	<p>Define the refinement facets.</p> <p>They will be used to filter search results and navigate through them. By default, they display a count of documents but you can add other calculations.</p> 	<p>Defining the Refinement Facets</p>
7	<p>Create the front end of your application with the Mashup Builder.</p>	<p>Creating the Front-end Application with Mashup Builder</p>

Step	Description	See ...
	<ul style="list-style-type: none"><li>• Use the Exalead CloudView feed to retrieve hits from Exalead CloudView and other feeds.</li><li>• Assemble page interfaces with widgets.</li><li>• Check your configuration in the Mashup UI.</li></ul> <p><b>Note:</b></p> <p>You can also create use your own tools to communicate with the Exalead CloudView Search API.</p>	
8	Optionally, secure the access to your documents.	<a href="#">Securing Document Access</a>

# Installing Exalead CloudView on Windows

This section explains how to quickly install and run the product on a Windows OS.

For a complete installation guide, covering system requirements, detailed installation steps on Windows and UNIX systems and deployment scenarios, see the *Exalead CloudView Administration Guide*.

## Install Exalead CloudView

To install Exalead CloudView you need the following files:

- The Exalead CloudView installation kit – download it or copy it from the DVD.
- A valid license file (`cvlicense.dat` for a standalone installation or a `.lic` file). To request a license, go to <http://www.3ds.com/terms/software-keys/>
- Any additional Exalead CloudView plug-ins (optional).

To do this task	You need these permissions
Start the Windows command line ( <code>cmd.exe</code> )	Administrator
Install Exalead CloudView	Administrator, plus: <ul style="list-style-type: none"> <li>• <b>Write</b> permissions for the installation directory, OR</li> <li>• <b>Create</b> permissions on the installation directory if it does not already exist</li> </ul>
Import files during setup	Read permission for the files to be imported.

1. Extract the Exalead CloudView archive to the target directory. This will create your `<INSTALLDIR>` folder, which will have the same name as the zip file.
2. Start the Window Command line (`cmd`) as **administrator**.
3. Run the installation script in `<INSTALLDIR>` with the following arguments:

```
install.bat -data <DATADIR> -license <LICENSE> -port
<BASEPORT> [-service]
```

Argument	Description
<code>&lt;DATADIR&gt;</code>	Creates the specified directory during installation, to store the index and configuration data for your Exalead CloudView instance. Specify a new directory for <code>&lt;DATADIR&gt;</code> and the installer will create it. For ease of maintenance, keep the <code>&lt;DATADIR&gt;</code> path separate from the <code>&lt;INSTALLDIR&gt;</code> path.

Argument	Description
<LICENSE>	<p>The path to the license file (<code>cvlicense.dat</code>)</p> <p>When using Exalead CloudView within the 3D#EXPERIENCE platform, you must also upload a dedicated <code>.lic</code> file to the Dassault Systèmes License Server (DSLS) to manage user and token count.</p>
<BASEPORT>	<p>Specifies the starting port for port allocation. Exalead CloudView needs a range of 100 consecutive TCP ports starting from &lt;BASEPORT&gt;. For example, the Mashup UI is installed by default on port 10000 (known as the BASEPORT) and is accessible at: <code>http://&lt;HOSTNAME&gt;:&lt;BASEPORT&gt;</code></p>
service	<p>Runs Exalead CloudView as a Windows service. For this option, use the default values for the local system account OR specify the account name and password as arguments:</p> <pre>-service [-user &lt;USERNAME&gt; [-password &lt;USERPASSWORD&gt;]]</pre>

- At the command prompt, enter **compmgmt.msc** and press **Enter**.  
The **Computer Management** dialog box is displayed.
- Under **Services and Applications > Services**, search for `Exalead CloudView - cvdefault` (default Exalead CloudView instance name).
- Right-click and select **Start**.
- Go to [Configure Exalead CloudView Using the Setup Wizard](#) to complete the Exalead CloudView setup.

## Check the Installation Logs If Errors Occur

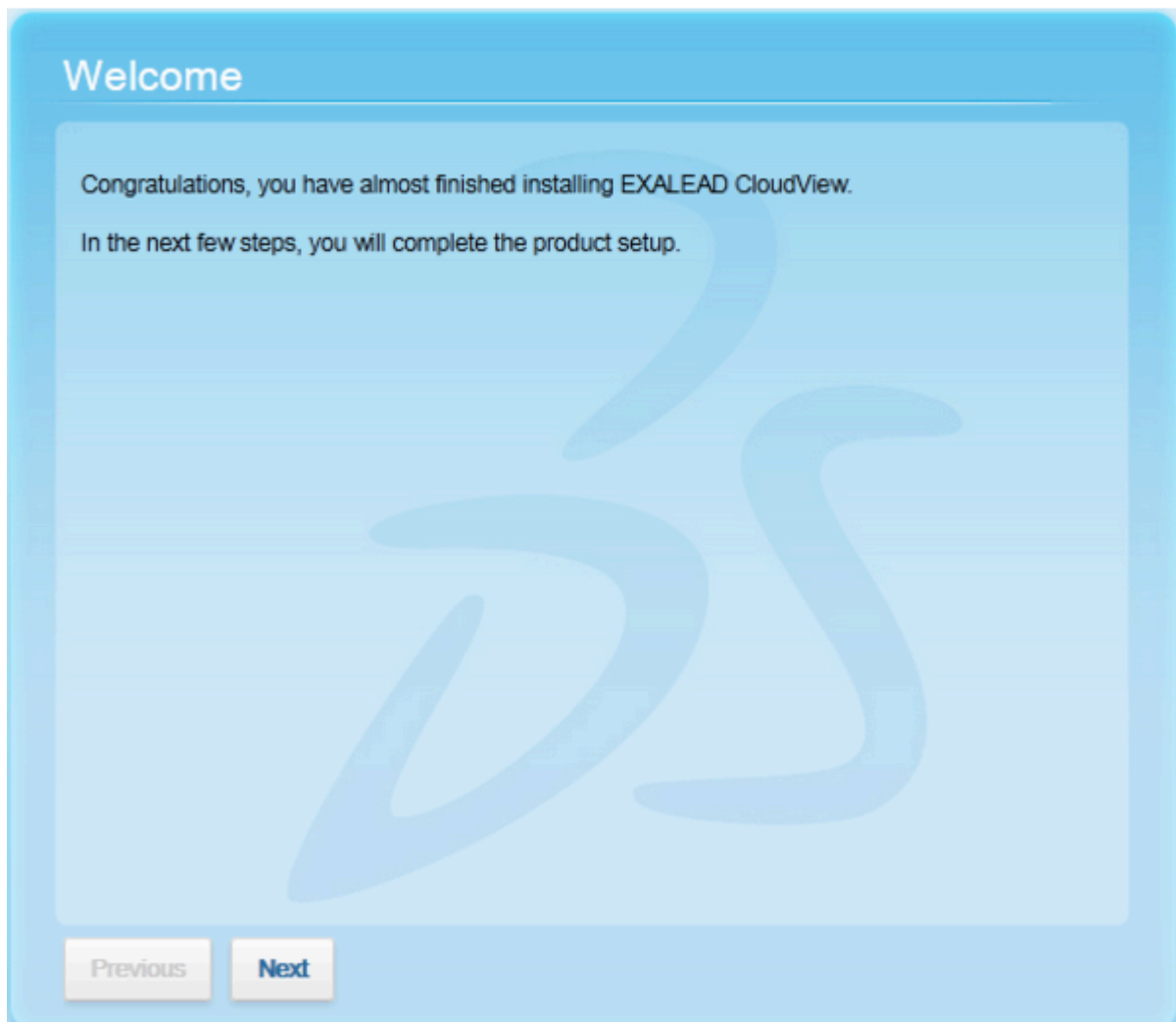
- If you encounter errors during the installation process, see the logs located in `<DATADIR>/run`.

## Configure Exalead CloudView Using the Setup Wizard

Complete the Exalead CloudView installation by performing some additional setup using the setup wizard.

- Go to `http://<HOSTNAME>:<BASEPORT+1>/setup`.

The **Welcome** screen opens.



2. On the **Welcome** screen, click **Next**.

The **Processing** screen opens.

**Processing**

**Consolidation**

- ☐ Setup a consolidation server role with standard configuration

**Semantic processing**

- ☐ Enable named entities detection
- ☐ Enable spell checking ("did you mean") and phonetic matching ("soundslike")
- ☐ Enable related terms

**Fonts for thumbnails**

- ☒ Download and install the True Type Fonts (TTF)
- ☐ Use the existing TTF directory

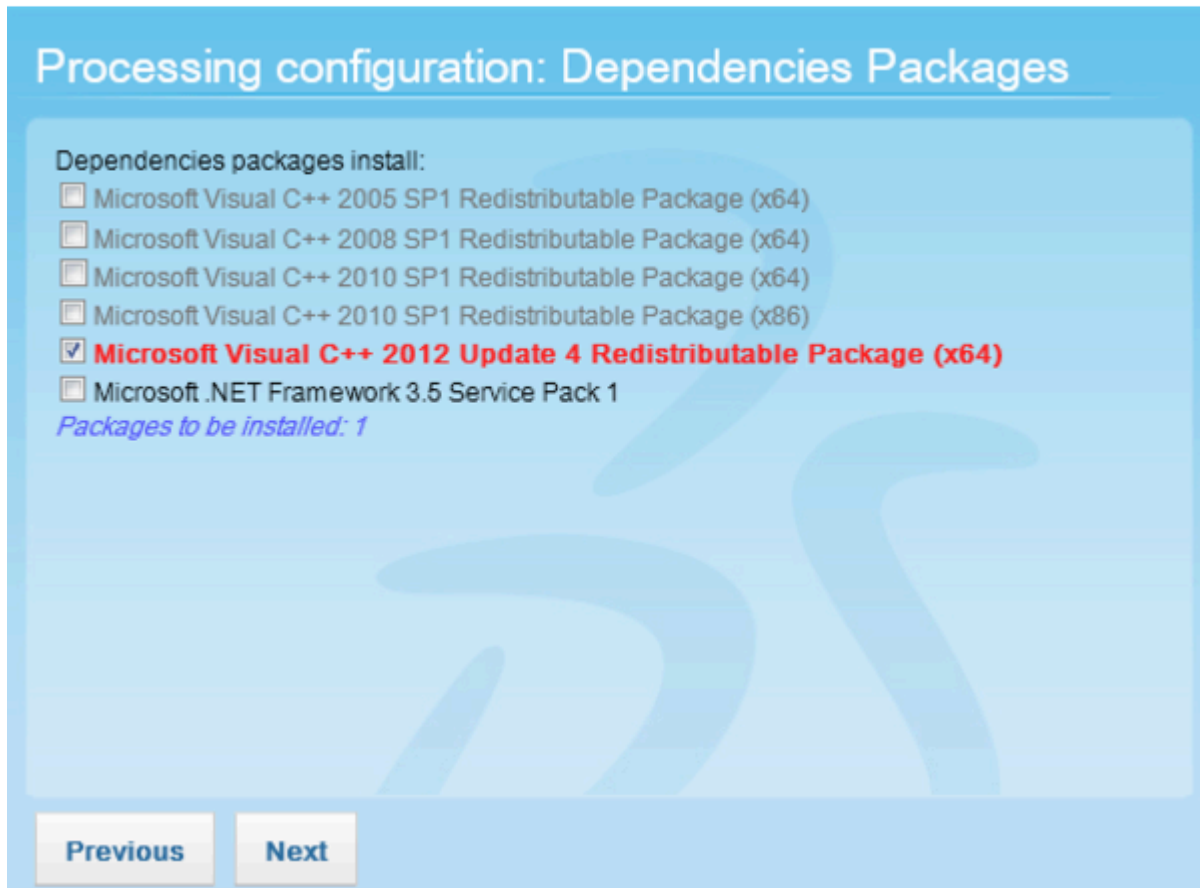
**Miscellaneous**

- ☒ Define metadata mappings for standard enterprise sources
- ☐ Install package dependencies (requires Administrator credentials)

[Previous](#) [Next](#)

3. Clear the **Define metadata mappings for standard enterprise** option to install Exalead CloudView with an empty document class.
4. Select **Install package dependencies**.
5. You will be taken to a separate screen to log in. Enter the Administrator user name and password, and then click **Next**.
6. As you have selected **Install package dependencies**, you will see a list of all dependency packages to be installed. Click **Next**.





7. On the **Administrator** screen, set up the options for:
  - **Account settings**: specify the login and password to use when accessing Exalead CloudView.
  - **Alerting** (optional) for critical Exalead CloudView events and license issues: specify the recipient email account settings.
  - **Reporting** (optional) on usage data and diagnostics: specify the proxy host and port.
  - Click **Next** when complete.

**Administrator**

**Account Settings (required)**

Login:

Password:

Reconfirm password:

**Alerting (optional)**

*Send email alerts for critical CloudView events and license token alerting*

Enable alerts: ☒

Recipient e-mail:

SMTP server host:

SMTP server port:

**Usage data (optional)**

*Help us identify trends and usage patterns by collecting usage data*

Collect anonymous usage data: ☒

*Internet connection proxy details (for usage data reporting and diagnostics)*

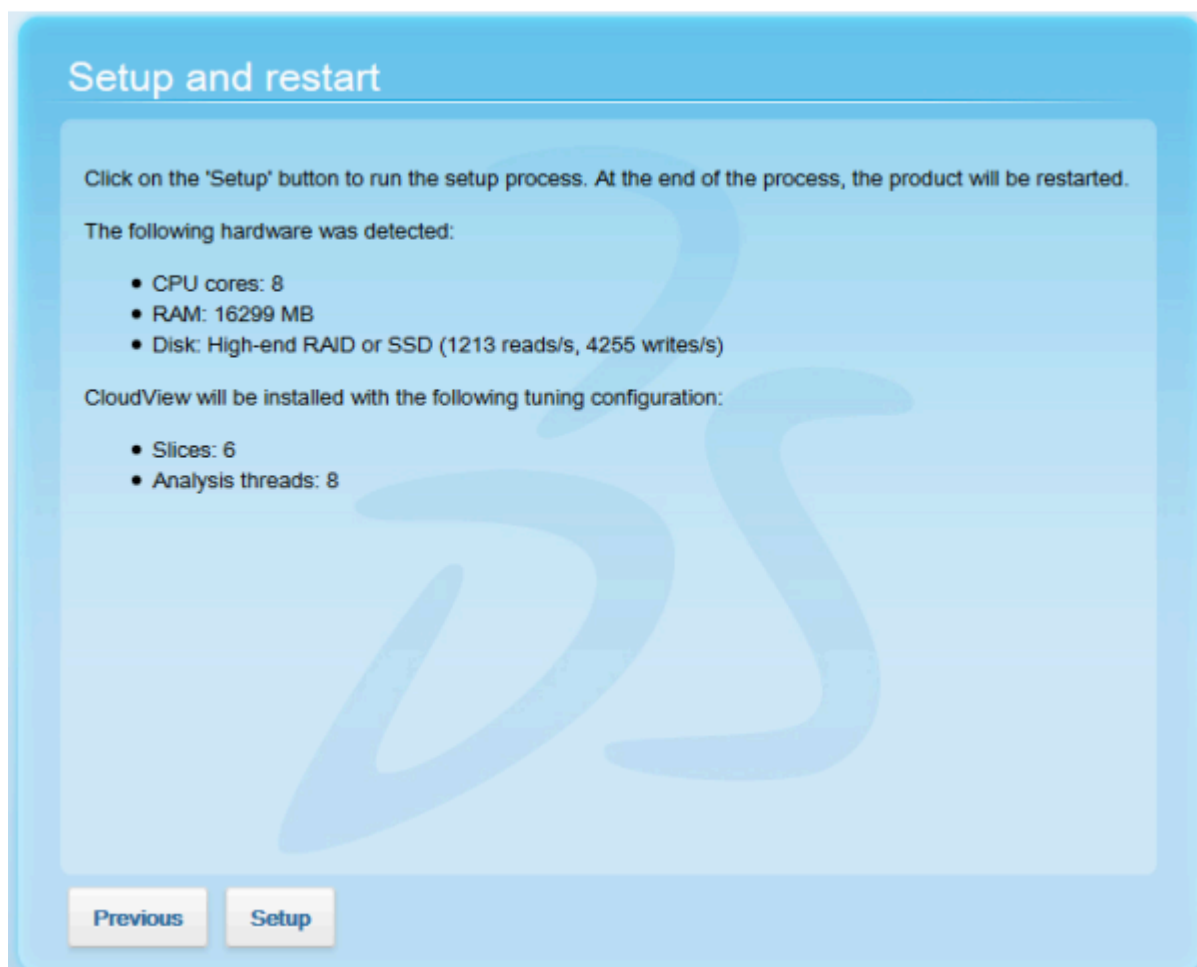
Proxy host:

Proxy port:

[Previous](#) [Next](#)

8. On the **Setup and restart** screen, click **Setup**.

This runs the setup process and then restarts Exalead CloudView automatically.



When you see the following screen, the setup is complete.

*Exalead CloudView Services page*

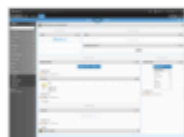


## BUILD APPLICATION



### Administration Console

Connect to data source, process, extract information and store it into the index.



### Mashup Builder

Build a search based application, configure business logic and result display.



### Business Console

Tune relevance, define content recommendation, edit semantic resources and alerting rules.

## GET HELP

### GUIDES

- Getting Started
- Installation & Deployment
- Connectors
- Application development
- Technical Reference

### REFERENCES

- Configuration
- Search parameters
- Virtual fields
- Mashup widgets
- Semantic resources

### CUSTOMIZE

- Connectors & Push API
- Customization SDK
- Java APIs SDK
- Mashup UI SDK

### RELEASE

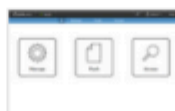
- Release notes
- Third-Parties

## MONITOR & TROUBLESHOOT



### Monitoring Console

Monitor index and search activity, hardware consumption and system status.



### API Console

Direct access to product configuration, manual document indexing...

## Access the Configuration Interfaces

Once the installation is complete, you can access the product interfaces and documentation through the Exalead CloudView Services available at `BASEPORT+1`, for example, `http://localhost:10001`.

The following interfaces are available:

Interface	Available at
Administration Console	<code>BASEPORT+1/admin</code>
Mashup Builder	<code>BASEPORT+1/mashup-builder</code>
Business Console	<code>BASEPORT+1/business-console</code>
Monitoring Console	<code>BASEPORT+1/perf-ui</code>
API Console	<code>BASEPORT+1/api-ui</code>

For more details, see [Exalead CloudView Software Interfaces](#).

1. From the Exalead CloudView Services page, select one of the configuration interfaces, for example, the **Administration Console**.

**Note:** You can also access it from: `http://<HOSTNAME>:<BASEPORT+1>/admin`.

2. Log in with the user name and password you specified in the Setup Wizard.

You now see the Administration Console **Home** page:

The screenshot displays the Exalead CloudView Administration Console Home page. The top navigation bar includes 'EXALEAD | CloudView', 'Save', 'Cancel', 'Apply', 'Go to...', a user profile 'admin', and a 'Help' button. The left sidebar lists various configuration categories. The main content area is divided into three primary sections: Connectors, Indexing, and Processes. The Connectors section features a table with columns for Name, Type, Status, Tasks, and Documents. A single connector named 'default' is listed with a status of 'n/a' and zero tasks or documents. The Indexing section includes a 'Build group' dropdown set to 'bg0' and buttons for 'Clear', 'Force commit', and 'Full compact'. It also shows a summary of indexing activities like Analysis, Import, and Compaction, all currently in an 'Idle' state. On the right side, two line graphs are present: 'Indexed documents on bg0' and 'Pushed tasks on bg0', both showing zero activity over a time period from 12:00 to 16:00. A link to the 'Monitoring Console' is provided for further data and history.

You can use the Administration Console for most configuration. However, the Exalead CloudView configuration is XML-based so, you can also edit the configuration manually or through the APIs. All changes made in the UI or the APIs will be reflected in the XML files.

# Creating a Simple Search Application

This chapter describes how to build an SBA following the recommended methodology.

This chapter is a tutorial following the plan presented in [Creating an Application \(Methodology\)](#).

For each section, we will describe the fundamentals of each methodology step, and use a sample database to illustrate them with examples.

This tutorial will also help you to familiarize with two main components of Exalead CloudView:

- The Administration Console to configure indexing and search.
- The Mashup Builder to configure how search results display in the Mashup UI.

[Defining the Application Scope](#)

[Adding a Connector](#)

[Creating the Data Model](#)

[Configuring the Search Bar Behavior](#)

[Preparing the Content to Display in the Search Results](#)

[Defining the Refinement Facets](#)

[Creating the Front-end Application with Mashup Builder](#)

## Defining the Application Scope

What do you want to search is the first question to ask yourself. You need to have a good understanding of the document corpus to configure your SBA properly.

You must know the type of data provided by your data sources:

- Unstructured data, for example, file system documents like pdfs, image files, Microsoft Office files, etc.
- Structured, for example, data coming from a JDBC database.

For each type of data source that will feed Exalead CloudView with data, what do you want to index? Do you want to index all data, or just a part of it?

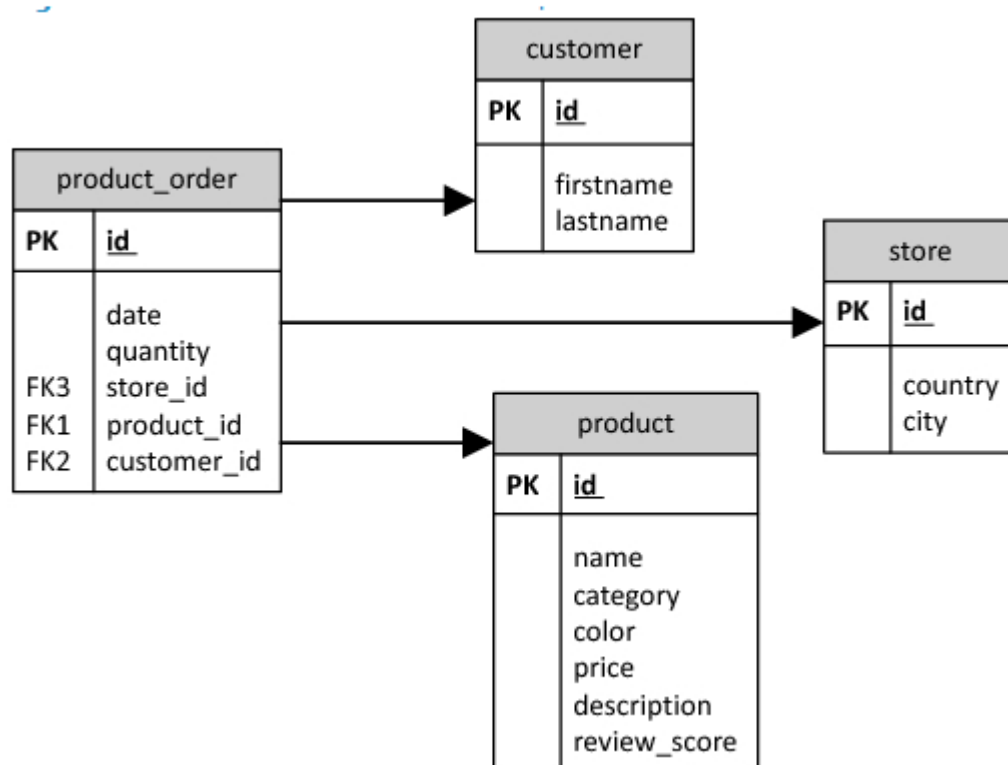
For example, if you index the content of a file system, do you want to exclude some files? For a database, do you want to retrieve all tables? What will be the primary keys in your table to give a meaning to your data?

## Tutorial: What Is Our Data Source

For this tutorial, we will use the sample SQLite database located in: `<INSTALLDIR>/docs/sample_database/data.db`

We thus have structured data coming from a database with four tables and the `id` field as primary key (PK) to make inner joins between them. The schema of the database is shown below.

*Database schema for the sample database*



## Tutorial: What We Want to Do Functionally

We want to use this database to create an SBA for an international clothing manufacturer. Its sales director wants it to be useful for various analytics needs, like showing the quantity of each article sold per city, the color of the most popular article in each country, the best salesmen, and saleswomen internationally and which articles they best sold to clothing stores.

## Adding a Connector

Once you have defined the scope of your application, you can add and configure connectors to fetch documents from data sources and push them into Exalead CloudView. Exalead CloudView will convert these original documents into Exalead CloudView documents.

For details on using a Database connector and other connectors packaged by default with Exalead CloudView, see "Configuring the JDBC Database Connector" in the *Exalead CloudView Connectors Guide*.

The following tutorial explains how to connect to the sample database.

1. In the Administration Console, select **Index > Connectors** and then click **Add connector**.
  - a. For **Name**, type `salesdb`.
  - b. For **Type**, select **Database (JDBC)**.
  - c. Click **Accept**.

2. For **Store documents in data model class**, enter `sales`. We will create this class later on.

3. In the **Connection parameters** section, complete the following:

- a. **Driver:** `org.sqlite.JDBC`
- b. **Connection string:** `jdbc:sqlite://<PATH TO INSTALLDIR>/docs/sample_database/data.db`

4. Click **Test connection**.

The database connector automatically connects to the database.

5. In the new **Query parameters** section that displays, enter the following in **Initial query**:

```
SELECT po.id as id, datetime(po.date, "unixepoch") as date,
po.quantity as quantity, p.price as unit_price, s.country as
store_country, s.city as store_city, c.firstname as firstname,
c.lastname as lastname, p.review_score as review_score, p.description
as description, p.color as color, p.category as category, p.name as
name

FROM product_order po inner join product p on p.id = po.product_id
inner join store s on s.id = po.store_id inner join customer c on c.id
= po.customer_id;
```

6. Click **Retrieve fields**.

We only need to set a field as being the primary key. The connector will use it to identify each record, and construct the Exalead CloudView document URL.

7. Set the `id` field as primary key.
  - a. Click the `id` field to expand it.
  - b. Select **Use as primary key**.
8. Click **Save**.

You are now ready to modify the data model.



## Creating the Data Model

This section gives a quick description of the data model concepts you must know to create a Search Application and a simple example of data model class creation.

[The Data Model in a Nutshell](#)

[Configuring Data Model Properties](#)

[Create a New Class in the Data Model](#)

### The Data Model in a Nutshell

This section gives you the basics to understand how the Data Model works.

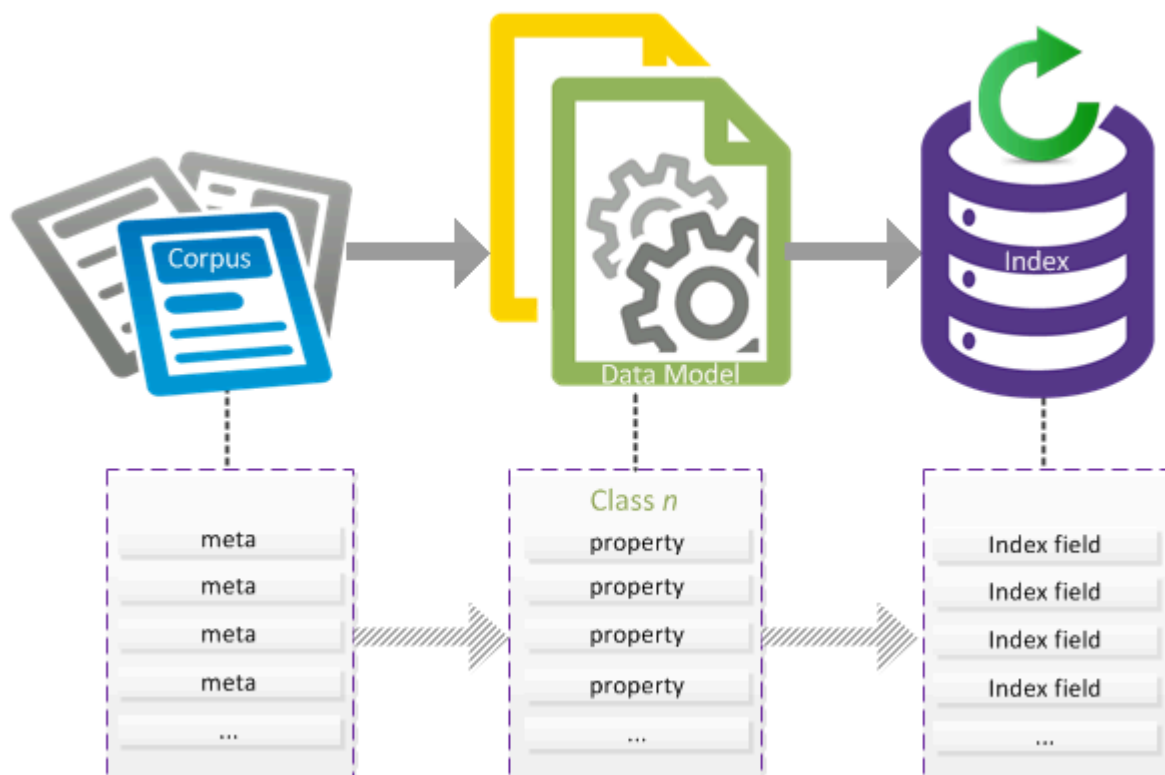
The **Data Model** defines how documents are stored in the Exalead CloudView index. It provides a way to map the relevant document metas available in your corpus to specific properties grouped by classes. The data model defines how data will be stored in the index, and how it will be used at search time.

The Data Model defines how items are stored in the Exalead CloudView index. It provides a way to gather the related metas found in your data sources into specific classes.

**Classes** allow you to group list of properties to clearly structure your data model according to the types of documents present in your corpus. You can consider them as business logic views. A document can belong to one class only.

**Properties** define the configuration of the document metas you want to index. During indexing, Exalead CloudView converts properties into index fields, and optionally hit display fields or facets, depending on your configuration.

*Transformation of document metas into index fields via the data model*



Exalead CloudView installs with a `default_model` data model containing a default `document` class but you can create your own data models and classes.

## Configuring Data Model Properties

This section explains the options available when defining Data Model properties.

### Choose a Data Type

First configure the type of each document meta you want to index.

For example, if the document meta contains:

- Textual values, select **alphanumeric**.
- Dates, select **Date (with time)** or **Date (without time)**.
- Numerical values, select **Integer**, **Double** or **Unsigned Integer**.
- Geographical coordinates, select **GPS point** or **XY point**.

### Assign Semantic Types to Alphanumeric Properties

When adding a new alphanumeric property, you must assign it a semantic type. It defines the global semantic processing to apply when indexing document metas. If required, you can then customize this semantic processing at the Analysis pipeline level (**Index > Data Processing > Semantic Processors**).

Exalead CloudView includes default semantic types, but you can modify them or create additional ones in **Data Model > Semantic Types**.

### Default Semantic Types

Semantic type	Description
<b>text</b>	<p>Apply this semantic type to document metas containing several words or sentences.</p> <p>For example, you would apply this semantic type for a meta with a value like:</p> <p>“Deep Blue was a chess-playing computer. It is known for being the first piece of artificial intelligence to win both a chess game and a chess match against a reigning world champion under regular time controls.”</p> <p>Also use this semantic type to perform extra processing treatment in the analysis pipeline.</p>
<b>metadata</b>	<p>Apply this semantic type to document metas containing a few words that do not require language detection and spell-checking.</p> <p>For example, if we have the meta value <code>Deep Blue</code></p> <p>Searching for <code>deep blue</code> in lowercase would work, as well as searching for <code>deep</code> only or for <code>blue</code> only.</p>
<b>identifier</b>	<p>Apply this semantic type to document metas containing IDs.</p> <p>With this semantic type, document metas will be searched in their normalized forms, but they will not be tokenized and the language will not be detected.</p> <p>For example, if we have the meta value <code>Deep Blue</code></p> <p>Searching for “<code>deep blue</code>” (with quotes) would work, but searching for <code>deep</code> only or <code>blue</code> only would not work.</p>
<b>url</b>	<p>Apply this semantic type to document metas containing web addresses.</p>

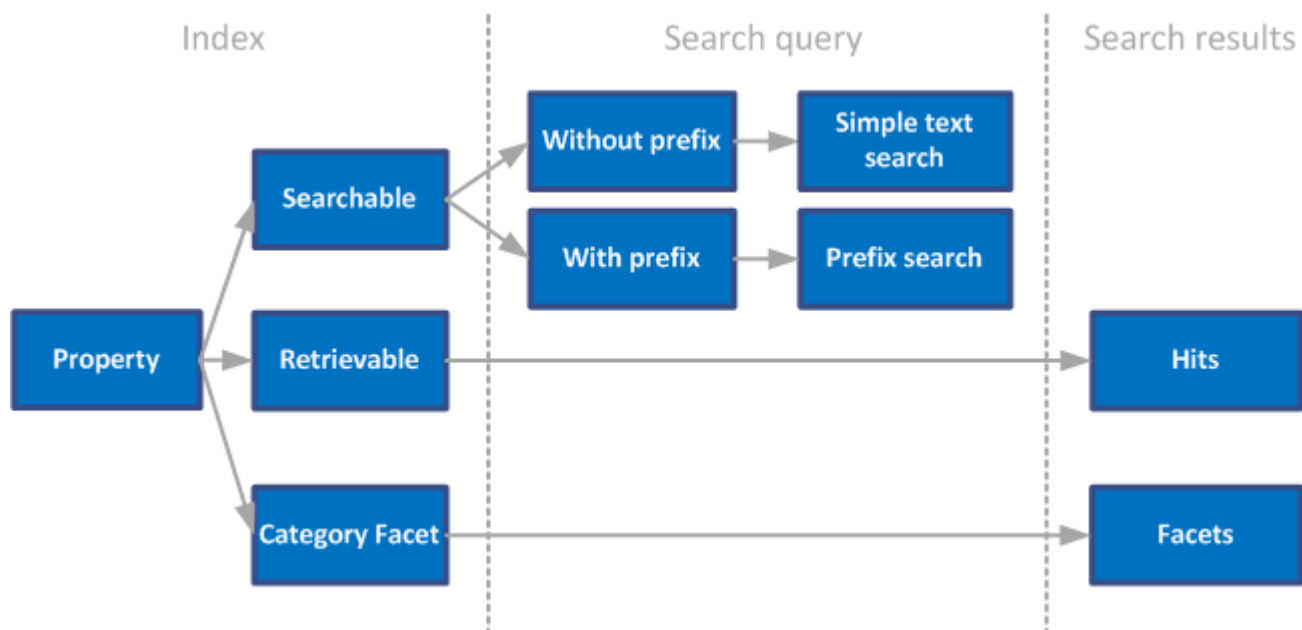
### Indexing Options for Data Model Properties

When creating a data model property, you must specify whether it will be either an index field or a facet, or both.

- **Dedicated field:** saves the meta as a dedicated field in the index, similar to a database field.
- **Category facet:** saves the meta in specific index field called categories, used by facets.

This section explains the options available depending on the property type. It can be globally represented by the following schema.

## Indexing options for data model properties



Properties have the following options:

- **Searchable with prefix:** allows users to search for matches in this particular meta, using a prefix. For example, to search for “API” in the title of a document, enter the search query `title:API`.
- **Searchable without prefix:** allows users to perform a simple text search, without using a prefix. For example, to search for “API” in a document, enter the search query `API`.
- **Retrievable:** enables this meta to be displayed in the hit content of search results.
- **RAM based:** stores this meta in RAM. Use this option if you need to sort search results on this meta, or when using this meta as a virtual facet.

**Note:** Property options may impact performance. For more details, see "How property options impact performance" in the *Exalead CloudView Configuration Guide*.

## Create a New Class in the Data Model

We want to configure the data model required to index the database fields. To do so, we will first create a new data model class.

### Tutorial: Add a New Class

1. In the Administration Console, select **Index > Data Model**.
2. In the **Classes** panel, click **Add class**:
  - a. Name it `sales`

- b. Select **no parent class**
- c. Click **Accept**

### Tutorial: Add Properties to the New Class

Once the class is created, we must configure a set of properties to map document metas to index fields and facets. To help us configuring these properties, we will use the **Trace all metas** option, which retrieves the list of metadata associated with the connector's documents.

1. In the **Classes** panel, select **Trace all metas** and click **Apply**.
2. Go to the **Home** page, click **Scan** for the `salesdb` connector.  
Once done, 15,000 documents are indexed.
3. Go back to **Index > Data Model > Classes**.
4. Select the `sales` class and click **Add properties from traced metas**.
5. Select the metas to be generated as properties and change their types according to the table below:

#### *Details for the sales class properties configuration*

Name	Data Type	Semantic Type	Field Type
<b>category</b>	Alphanumeric	metadata	Category facet
<b>color</b>	Alphanumeric	metadata	Category facet
<b>description</b>	Alphanumeric	text	Dedicated field
<b>firstname</b>	Alphanumeric	text	Dedicated field
<b>id</b>	Alphanumeric	metadata	Dedicated field
<b>lastname</b>	Alphanumeric	text	Dedicated field
<b>name</b>	Alphanumeric	text	Dedicated field
<b>quantity</b>	Integer	n/a	Dedicated field
<b>store_city</b>	Alphanumeric	metadata	Category facet
<b>store_country</b>	Alphanumeric	metadata	Category facet
<b>unit_price</b>	Double	n/a	Dedicated field

Set your properties as follows:

**Add properties from traced metas**

Filter by meta:  Filter by source:

<input type="checkbox"/>	Meta	Frequency	Data type	Semantic type	Multivalued	Dedicated field	Category facet
<input type="checkbox"/>	<a href="#">analysisdate</a>	1051	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">category</a>	1055	Alphanum	metadata	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">color</a>	1055	Alphanum	metadata	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<a href="#">dataModelClass</a>	1051	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">dataModelClassHierarchy</a>	1051	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">date</a>	50	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">description</a>	1053	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">firstname</a>	483	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">id</a>	50	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">lastname</a>	446	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">name</a>	1053	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">quantity</a>	905	Integer		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">review_score</a>	1053	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">source</a>	1051	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">source_path</a>	1051	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">store_city</a>	1053	Alphanum	metadata	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">store_country</a>	1053	Alphanum	metadata	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<a href="#">unit_price</a>	1053	Double		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<a href="#">uri</a>	50	Alphanum	text	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

1-19 of 19

[Generate properties](#) [Cancel](#)

6. Click **Generate properties**.

The new properties appear for the `sales` class.

The screenshot shows the 'Classes' tab in the Exalead CloudView interface. The 'sales' class is selected, and its properties are being configured. The 'Properties for the sales class' section shows a list of properties with checkboxes for 'Dedicated field', 'Dynamic field', and 'Category facet'. The 'Dynamic properties for the sales class' section is empty.

Name	Type	Dedicated field	Dynamic field	Category facet	Actions
category	metadata (Alphanumeric)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
color	metadata (Alphanumeric)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
description	text (Alphanumeric)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
firstname	text (Alphanumeric)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
lastname	text (Alphanumeric)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
name	text (Alphanumeric)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
quantity	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
store_city	metadata (Alphanumeric)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
store_country	metadata (Alphanumeric)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
unit_price	double	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

7. You must edit the properties as detailed below.

Property	Data Type	Option to add
description	Alphanumeric	Searchable without prefix
name	Alphanumeric	Searchable without prefix

8. Clear the **Trace all metas** option.

9. Click **Apply**.

### Tutorial: Re-Index Data Using the New sales Class

To map document metas to the **sales** class properties, we need to clear what has been indexed previously and reindex data.

1. On the **Home** page, click **Clear documents** for the `salesdb` connector.
2. Click **Accept** to confirm.

Wait for the index to clear its documents.

3. Click **Scan** for the `salesdb` connector.

After a few seconds, records are pushed and Exalead CloudView processes them. Once the scan is complete, there are 15,000 indexed documents.

## Tutorial: See the Search Results

Now that data has been indexed, you can check the results of what has been configured so far on the default front end.

1. Go to the Mashup UI: <http://<HOSTNAME>:<BASEPORT>/mashup-ui>
2. For our example, search for all documents by entering `#all` in the search field.

The hit metas and hit facets you can see in each hit content, are displayed because the **retrievable** option is selected by default when you add properties.

The orange rectangle, displays the hit metas. The blue rectangles display the metas configured as category facets and hit facets.

The screenshot shows the Mashup UI search results page. At the top, a blue bar indicates 'Results 1-10 of 15000' and 'Sort by Relevance Date Size'. The main content area displays two document hits. The first hit, with ID 'id=9441&', is highlighted with an orange rectangle. The second hit, with ID 'id=1235&', is highlighted with a blue rectangle. To the right, a 'Refinements' sidebar shows various filters and their counts.

**Hit 1 (id=9441&):**

category	trousers	color	black
description	Soft silk trousers. High waisted with concealed hook & eye fastening. Outer Material: 50% Viscose, 50% Acetate. Two side pockets, two back jetted pockets.		
lastname	Salas	name	Luxury trousers
quantity	1	store_city	London
store_country	UK	unit_price	159.99

**Hit 2 (id=1235&):**

category	shorts	color	black
description	Slim fit. 57% Cotton, 4% Elastane, 39% Nylon (polyamide). Cool iron. Machine wash at max 40°C. Do not bleach. Do not dry clean.		
lastname	Holden	name	Smart slim shorts
quantity	1	store_city	London

**Refinements:**

- sales\_store\_country**
  - USA (3304)
  - France (2977)
  - UK (2017)
  - Germany (2308)
  - Brazil (1383)
  - Japan (1371)
  - China (881)
  - Australia (111)
  - Canada (88)
- sales\_color**
  - black (8228)
  - white (3010)
  - red (2613)
  - multiple (2336)
  - blue (807)
- sales\_category**
  - t-shirt (4047)
  - trousers (2798)
  - shirt (2499)
  - swimwear (2336)
  - shorts (1120)
  - belt (1111)
  - hat (1083)
- Source**
  - salesdb (15000)
- Language**
  - English (15000)
- Data model class**
  - sales (15000)
- sales\_store\_city**
  - London (15000)



## Configuring the Search Bar Behavior

Once the data model is created, you can specify the behavior of your application search bars, that is to say what will your end-users be able to do when entering search queries, how will you help them to correct their queries or to choose a relevant query.

### Use Full-Text Search

### Use Prefix Search to Make Specific Queries

### Use Semantic Analysis to Correct Queries

### Use Full-Text Search

We call full-text search, a user query that does not contain any prefixes. By default, when you enter a query, Exalead CloudView searches the content of the `text` index field.

In addition to their specific index fields, all properties set to **searchable without prefix**, also send their content to the `text` index field. This allows users to search this property without specifying a prefix in the query.

For example, `cropped trousers` is a simple query for which Exalead CloudView will search for the term `cropped`, the term `trousers`, and the combination of these two terms.

The following tutorial shows how to use full-text search with Boolean operators

You are certainly familiar with common query operators, like:

- Quotes `"` to search for exact phrases. For example, `"cropped trousers"` will search for the exact expression, not for `cropped` and `trousers` separately.
- `OR` to search for either one term or another. For example, `cat OR pet`.
- `AND` to search for one term and another term.

**Note:** Blank spaces are interpreted as `AND` implicitly.

- And maybe also `-` (minus sign) to exclude a term from the search. For example, `new -york` will search for documents containing `new` but not `york`.

Exalead CloudView provides these operators and many others in its User Query Language (UQL), the natural language used for queries. You can use these operators to make simple or rich queries by combining them. For more information, see "User Query Language (UQL)" in the *Exalead CloudView Configuration Guide*.

1. Go to the Mashup UI: `http://<HOSTNAME>:<BASEPORT>/mashup-ui`
2. In the Search field, enter: `trousers OR shirt NOT "t-shirt"`

Search results display hits for occurrences of trousers and shirt found in the `name` and `description` fields (which we set to **searchable without prefix** in [Create a New Class in the Data Model](#)).

The screenshot shows the EXALEAD search interface. The search bar contains the query "trousers OR shirt NOT 't-shirt'" and a "Search" button. Below the search bar, a blue bar indicates "Results 1-10 of 4388" and sorting options: "Sort by Relevance Date Size".

The results are displayed in a table-like format. The first result (id=8639) shows the following details:

category	trousers	color	red	
description	Very comfortable cropped trousers. High waisted with concealed hook and eye fastening. Concealed zip. Side pockets and rear welt pockets.		firstname	Dominic
lastname	Knapp	name	Cropped trousers	
quantity	1	store_city	Tokyo	
store_country	Japan	unit_price	39.99	
sales_store_country	Japan	sales_color	red	
sales_category	trousers	Source	salesdb	
Data model class	sales	Language	English	
sales_store_city	Tokyo			

The second result (id=11913) shows similar details, but with a different first name (MacKensie) and a different description (Very comfortable cropped trousers. High waisted with c...).

On the right side, there is a "Refinements" sidebar with the following sections:

- sales\_store\_country**: USA, France, UK, Germany, Japan, Brazil, China, Australia, Canada.
- sales\_color**: red, black, blue.
- sales\_category**: shirt (highlighted), trousers.
- Source**: salesdb.
- Language**: English.
- Data model class**: sales.
- sales\_store\_city**: Lyon, Paris.

## Use Prefix Search to Make Specific Queries

**Prefix handlers** allow you to refine queries by targeting specific index fields or change the behavior of the query expansion. For example, the `title:` prefix handler allows you to refine the search on document titles.

Prefix handlers are created automatically when you add data model properties.

This tutorial explains how to use prefix search (using prefix handlers) to target specific index fields. This gives us a means to search for data very precisely. For example, we can search for all articles with a given price using the `sales_unit_price` prefix (where `sales_` is the parent class of the `unit_price` property).

1. Go to the Mashup UI: `http://<HOSTNAME>:<BASEPORT>/mashup-ui` .
2. In the Search field, enter: `sales_unit_price<15`






All the articles with a price lower than 15 are displayed.

The screenshot shows the EXALEAD Mashup UI interface. At the top, the EXALEAD logo is on the left, and a search bar contains the query `sales_unit_price<15`. Below the search bar, a blue header bar displays 'Results > 1-10 of 4098' and a 'Sort by' dropdown menu with options: Relevance, Date, and Size. The main content area shows a document preview for `id=1235&`. The document is a JSON object with the following fields and values:

<code>category</code>	<code>shorts</code>	<code>color</code>	<code>black</code>
<code>description</code>	Slim fit. 57% Cotton, 4% Elastane, 39% Nylon (polyamide). Cool iron. Machine wash at max 40°C. Do not bleach. Do not dry clean.		
<code>lastname</code>	<code>Holden</code>	<code>name</code>	<code>Smart slim shorts</code>
<code>quantity</code>	<code>1</code>	<code>store_city</code>	<code>London</code>
<code>store_country</code>	<code>UK</code>	<code>unit_price</code>	<code>14.99</code>
<code>sales_store_country</code>	<code>UK</code>	<code>sales_color</code>	<code>black</code>
<code>sales_category</code>	<code>shorts</code>	<code>Source</code>	<code>salesdb</code>
<code>Data model class</code>	<code>sales</code>	<code>Language</code>	<code>English</code>
<code>sales_store_city</code>	<code>London</code>		

At the bottom of the document preview, there is a link: `id: id=1235&`.

3. `sales_unit_price` is not very handy for our prefix search, and we want to be able to use this prefix handler with `price` only.
  - a. In the Administration Console, go to **Search > Search Logics > Query Language**.
  - b. Expand the `sales_unit_price` prefix handler.
  - c. Click **Customize**.
  - d. In the **Aliases** field, add `price` as new alias. If you want to enter several aliases, separate them by commas **WITHOUT** space.

sales_unit_price(price)		Numeric
Index field	sales_unit_price	
Default operator	= (equals)	
Invalid value behavior	Error	
Retrieved field	<input type="checkbox"/>	
Set as default handler	<input type="radio"/>	
Aliases	price	

e. Click **Apply**.

f. In the Mashup UI, you can now search for `price: 14.99`

4. Now let us combine prefix search with Boolean operators. Enter the query:

```
sales_firstname: Dean sales_lastname: salas shirt NOT "t-shirt"
```

You will see all shirts sold by Dean Salas, but not t-shirts.

## Use Semantic Analysis to Correct Queries

To be able to provide relevant search results when the user's query is incomplete, misspelled, or imprecise, Exalead CloudView performs a semantic analysis of documents as well as the queries themselves. This generates word matching operators and fuzzy matching options.

For example:

- Did you mean? Spell check: `exalaed` will prompt `Did you mean: exalead?`.
- Approximation: `exalaed` will match `exalead`.
- Phonetic spelling: `exaleed` will match `exalead`.
- Word truncations: `exal*` will match `Exalead`, `exalid` `exalted`.
- Regular expressions: `/exa.ead/` will match `exalead` and `exahead`.

For more information, see "More About Semantic Analysis" in the *Exalead CloudView Configuration Guide*.

Depending on the semantic feature, the analysis takes place either at indexing-time, or at search-time.

- Index-time: analyzes documents just before indexing, using semantic processors. Anytime you modify semantic processors, you must always reindex your documents before the change will appear in your application.

- **Search-time:** analyzes the user's search request, known as query expansion, which essentially adds additional search terms to the user's original query. For example, if phonetic query expansion is enabled, the query `exaleed` will be expanded to `"exaleed" OR "exalead"`.

**Note:** The following tutorial example explains how to configure semantic processing at search time. We will enable approximation for full-text search, to correct user queries with typos. For example, if the user enters `cropped`, the search results will display hits with `cropped`, the correct spelling, automatically. To enable approximation, we must modify the query expansion configuration for the `text` prefix. For details on configuring semantic processing at index-time, see the *Exalead CloudView Configuration Guide*.

## Search Without Approximation

1. Go to the Mashup UI: `http://<HOSTNAME>:<BASEPORT>/mashup-ui` .
2. In the search bar, enter: `cropped` and click **Search**

Records do not display in the search results as the word is misspelled.

## Enable Approximation

1. In the Administration Console, go to **Search > Search Logics > sl0 > Query Expansion**.
2. Under **Query Expansion Modules**, verify that an **approximate** module appears in the list. If not, add one by clicking **Add module**.
3. Go to the **Query language** tab.
4. Click the **text** prefix handler.
  - a. Beside **Query expansion config**, click **Edit**.
  - b. In the **Query expansion config** window, double-click the **approximate** module.
  - c. Click **Accept**.

text - default		Text
Target index fields	<input type="text" value="text"/>	<a href="#">i</a>
Index fields weights	<input type="text"/>	<a href="#">i</a>
Expand end of expression	<input type="checkbox"/>	<a href="#">i</a>
Use indexed pattern search	<input type="checkbox"/>	<a href="#">i</a>
Use nested prefix as dyn. meta	<input type="checkbox"/>	<a href="#">i</a>
Dyn. meta name	<input type="text"/>	<a href="#">i</a>
Target dictionary	<input type="text" value="dict0"/>	<a href="#">i</a>
Query expansion config	<input type="text" value="approximate{"/> <a href="#">Edit</a>	<a href="#">i</a>
Set as default handler	<input checked="" type="radio"/>	
Aliases	<input type="text"/>	<a href="#">i</a>

5. Click **Apply**.

## Search with Approximation

1. In the Mashup UI, search for `cropped`.

This time, records are displayed in the search results, since Exalead CloudView expanded the query with approximation to get the correct orthograph for all fields that have a `text` semantic type. This is the case for the `description` and `name` fields in our example.

Results › 1-10 of 921		Sort by Relevance Date Size	
id=28&		Download	Preview
category	trousers	color	red
description	Very comfortable <b>cropped</b> trousers. High waisted with concealed hook and eye fastening. Concealed zip. Side pockets and rear welt pockets.	firstname	Leonard
lastname	Kennedy	name	<b>Cropped</b> trousers
quantity	2	store_city	London
store_country	UK	unit_price	39.99
Source	sales	Data model class	sales
sales_store_city	London	Language	English
sales_color	red	sales_category	trousers
sales_store_country	UK		
id: id=28&			

## Preparing the Content to Display in the Search Results

A search result often called *hit* corresponds to a document in the index that contains a match to the user query.

The **Search Logics > Hit Content** tab allows you to define the default content of your search result pages:

- The number of hits to display on search result pages through the **Number of hits retrieved** option.
- The metas to display for each hit.
- The post-processing to apply for each hit:
  - Display a summary of the hit.
  - Highlight matching terms in its content.

Select the Hit Metas and Hit Facets to Display

Highlight Query Terms in Search Results

## Summarize the Content of Hit Metas

### Select the Hit Metas and Hit Facets to Display

When you go to the Administration Console > **Search > Search Logics > Hit content** and **Facets** tabs, the data model expansion has created the content to display in the search results automatically.

The hit metas and hit facets listed here correspond to all the properties that were set as **retrievable**. They are all prefixed by `sales_` as they belong to the **sales** class.

**Note:** The use of navigation facets is described separately, in [Defining the Refinement Facets](#). This tutorial explains how to enhance the display of hit facets.

As we only have one language in our corpus, we are going to remove the **Language** facet from both the hit content and the navigation facets (**Refinements**). We also want to hide from the hit content the facets:

- **sales\_category**
- **sales\_color**
- and **sales\_store\_country**

... as they are already displayed as hit metas (they were set as **retrievable** in the data model configuration) and we can refine on them using the **Refinements** panel.

1. In the Administration Console, go to **Search > Search Logics > sl0 > Facets**.

*The **Facets** tab shows the category facets that will be displayed as hit facets and navigation facets in the search results*



Select the Hit Metas and Hit Facets to Display

< **Facets** Hit Content Sort & Relevance Query Language Query Expansion Limits Query Template V >

Facets allow you to categorize documents based on their content. Facets can then be used to filter or "refine" search results.

**Facets** ? Add facet

Name <input type="text" value="Filter..."/>	Type <input type="text" value="Filter..."/>	Actions
<u>dataModelClass</u>	Category <input checked="" type="checkbox"/> Use for hit content <input type="checkbox"/> Use for navigation	X
<u>dataModelClassHierarchy</u>	Category <input type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	X
<u>extracted_numpages</u>	Category <input checked="" type="checkbox"/> Use for hit content <input type="checkbox"/> Use for navigation	X
<u>Source</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	X
<u>Language</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	X

**Facets from data model**  
You can drag and drop these facets in the list above to customize them


<u>sales_category</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	
<u>sales_color</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	
<u>sales_store_city</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	
<u>sales_store_country</u>	Category <input checked="" type="checkbox"/> Use for hit content <input checked="" type="checkbox"/> Use for navigation	

2. Delete the **Language** facet.
3. To hide the **sales\_category**, **sales\_color**, **sales\_store\_city** and **sales\_store\_country** facets:
  - a. Expand the **category** facet.
  - b. Click **Customize**.
  - c. Clear **Use for hit content**.
  - d. Repeat the previous steps for the other facets.
4. Click **Apply**.
5. Go to the Mashup UI search page: `http://<HOSTNAME>:<BASEPORT>/mashup-ui /page/search?q=`
6. Refresh the view.

The **Language** facet has been removed from the hit content and the **Refinements** panel. The **sales\_category**, **sales\_color**, **sales\_store\_city** and **sales\_store\_country** facets have also been removed from the hit content.


Results › 1-10 of 15000

Sort by Relevance Date Size

 id=9441&

category	trousers	color	black
description	Soft silk trousers. High waisted with concealed hook & eye fastening. Outer Material: 50% Viscose, 50% Acetate. Two side pockets, two back jetted pockets.	firstname	Dean
lastname	Salas	name	Luxury trousers
quantity	1	store_city	London
store_country	UK	unit_price	159.99
Source	salesdb	Data model class	sales

id: id=9441&

 id=1235&

category	shorts	color	black
description	Slim fit. 57% Cotton, 4% Elastane, 39% Nylon (polyamide). Cool iron. Machine wash at max 40°C. Do not bleach. Do not dry clean.	firstname	Destiny
lastname	Holden	name	Smart slim shorts
quantity	1	store_city	London
store_country	UK	unit_price	14.99
Source	salesdb	Data model class	sales

id: id=1235&

## Highlight Query Terms in Search Results

A common search application requirement is to display the results with the matching terms highlighted, in bold or in a different color.

**Note:** You can configure highlighting on any meta, but it is mostly applied on the `text` and `title` metas, as they allow users to quickly identify whether the hit is interesting or not.

### Tutorial: Configure the Highlighting of Query Terms for the Description Meta

1. In the Administration Console, go to **Search > Search Logics > sl0**.
2. On the **Hit Content** tab, click the **sales\_description** meta to display its settings.
3. Click **Customize**.
4. Select the **Highlight** check box.
5. Click **Apply**.

### Tutorial: Check the Highlighting of Query Terms

1. Go to the Mashup UI search page: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/page/search?q=`
2. Enter the query `nylon`

In the search results, **Nylon** is highlighted in bold in the **description** meta.

Results › 1-10 of 4335

Sort by Relevance Date Size

id=1235&

category	shorts	color	black
description	Slim fit. 57% Cotton, 4% Elastane, 39% <b>Nylon</b> (polyamide). Cool iron. Machine wash at max 40°C. Do not bleach. Do not dry clean.		
lastname	Holden	name	Smart slim shorts
quantity	1	store_city	London
store_country	UK	unit_price	14.99
Source	salesdb	Data model class	sales

id: id=1235&

### Summarize the Content of Hit Metas

A hit can contain a summary, which is a small number of selected sentences, centered around the query terms. You can typically configure its length and whether to highlight search terms in its content.

**Note:** You can apply a summary on any meta, but it is mostly applied on the `text` meta or on metas likely to contain rather long text.

### Tutorial: Configure a Summary for the Description Meta

In our tutorial, the **description** meta is the only one for which applying a summary is relevant.

The descriptions are not very long but let us say that we want to apply a summary to this meta value, to reduce the number of characters displayed and the overall height of the hits.

1. In the Administration Console, go to **Search > Search Logics > sl0**.
2. On the **Hit Content** tab, click the **sales\_description** meta.
3. Select the **Summary** check box.
4. Expand **Operations**, and expand **Summary**.
5. For our example, reduce the **Max. length** value to 40.
6. Click **Apply**.

### Tutorial: Check the Summary Display

1. Go to the Mashup UI search page: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/page/search?q=`
2. Refresh the screen.

In the search results, nylon is now highlighted in a summary of the **description** meta value.

Results › 1-10 of 4335

Sort by Relevance Date Size

id=1235&

category	shorts	color	black
description	57% Cotton, 4% Elastane. 39% Nylon ...	firstname	Destiny
lastname	Holden	name	Smart slim shorts
quantity	1	store_city	London
store_country	UK	unit_price	14.99
Source	salesdb	Data model class	sales

id: id=1235&

## Defining the Refinement Facets

Use navigation facets to filter search results and navigate through them. By default, they display in the **Refinements** panel of the Mashup UI, with a count of documents but you can use other calculations.

Navigation facets can also be displayed in charts and table widgets, where you can perform similar refinements. For example, when clicking a pie chart slice, the view is refined to focus on the slice details.

**Note:** For more information, see "Creating Facets to Refine Search Results" in the *Exalead CloudView Configuration Guide*.

### Tutorial: Enhance the Refinement Facets Display

We do not need the **Source** and **Data model class** facets in the **Refinements** panel, as we have only one source and one class. We want to hide them to reduce the number of facets.

1. In the Administration Console, go to **Search > Search Logics > sl0 > Facets**.
2. For the **dataModelClassHierarchy** facet, clear **Use for navigation**.
3. Repeat this action for the **Source** facet.
4. Click **Apply**.
5. Go to the Mashup UI search page: `http://<HOSTNAME>:<BASEPORT>/mashup-ui/page/search?q=`
6. Refresh the view.

The **Data model class** and **Source** facets have been removed from the **Refinements** panel.

### Tutorial: Add a Numerical Range facet for Unit Prices

For our tutorial, we are going to create a virtual numerical facet to organize search results by price ranges. We want to define a fixed size for all ranges, to show included units by ranges of 20 (and get \$1-21, 21-41, etc.).

To get more examples with other facet types, see [Tools for Analytics](#).

#### Create the Numerical Range Facet

1. In the Administration Console, go to **Search > Search Logics > Facets**.
2. Click **Add facet**.
  - a. Specify a **Name**, for example `price_range`

- b. For **Type**, select **Numerical ranges**.
  - c. Click **Accept**.
3. Clear **Use for hit content**, as we want this facet to display only for navigation in the **Refinements** panel.
4. For **Expression**:
  - a. Click **Edit**
  - b. Double-click `sales_unit_price` to insert it to the **Expression** field
  - c. Click **Accept**.
5. For **Sort by**, select **Range**.
6. For **Create ranges**, select **that are the same size**.
7. For **Range size**, enter 20.
8. We want to display price ranges separated by a dash (for example, \$21-41). By default, values are displayed between square brackets separated by semicolons, for example, [\$21;\$41]).
  - a. Expand **Range labels**.
  - b. Set **Range label format** to: \\$\$-\$

The facet configuration must be the following.

The screenshot shows the 'Facets' configuration panel for a facet named 'price\_range'. The 'Type' is set to 'Numerical ranges'. The 'Expression' is 'sales\_unit\_price'. The 'Default precision' is 0. The 'Sort by' is 'Range' with a 'Reverse' checkbox. The 'Create ranges' is set to 'that are the same size' with a 'Range size' of 20. The 'Range labels' section is expanded, showing the 'Range label format' set to '\\$\$-\$'. Other options like 'Above max range label format', 'Below min range label format', and 'Singleton range label format' are also visible. The 'Advanced options' section is collapsed.

9. Click **Apply**.

### Check the Range Facet Display

1. Go to the Mashup UI search page: <http://<HOSTNAME>:<BASEPORT>/mashup-ui/page/search?q=>
2. Remove `nylon` from the search bar and click **Search**.

In the search results, the **price\_range** facet is now displayed in the **Refinements** panel.

▼ price_range	
\$ 1-21	5778
\$ 21-41	6270
\$ 41-61	1984
\$ 141-161	968
▼ sales_store_city	
Lyon	1551
Paris	1426
Liverpool	1414
Rio de Janeiro	1383
Berlin	1275
London	1203
Munich	1033
Chicago	1031
Los Angeles	815
Tokyo	800
Washington D.C	601
Kyoto	571
San Francisco	424
Beijing	410

3. Click a range, for example **21-41**.

The results are refined to show hits with unit prices included within this range.

The screenshot displays the Mashup Builder interface. On the left, two data model classes are shown, both of type 'sales'. The top class has filters for color (multiple), firstname (Ginger), name (Bikini bottoms), store\_city (Paris), and unit\_price (34.99). The bottom class has filters for color (black), firstname (Wanda), name (Fedora hat), store\_city (Liverpool), and unit\_price (24.99). On the right, the 'Refinements' panel shows a hierarchical list of filters with their respective counts. The 'sales\_store\_country' filter is expanded, showing counts for USA (1383), France (1289), UK (1078), Germany (945), Brazil (588), Japan (554), China (360), Australia (50), and Canada (25). The 'sales\_color' filter is expanded, showing counts for multiple (2338), black (2194), and red (1740). The 'sales\_category' filter is expanded, showing counts for swimwear (2338), belt (1111), hat (1083), trousers (921), and shirt (819). The 'price\_range' filter is expanded, showing a refined range of '\$ 21-41' with a count of 6270. The 'sales\_store\_city' filter is expanded, showing counts for Lyon (872), Paris (817), Rio de Janeiro (588), Liverpool (572), and Berlin (516).

To see all ranges again, click the refined range.

## Creating the Front-end Application with Mashup Builder

The interface that end users use to search the index is known as a "search application". Exalead CloudView includes a default search application known as the Mashup UI, and a tool to modify it, the Mashup Builder.

For this tutorial, we will use the Mashup Builder to modify the default Mashup UI.

[About Mashup Builder](#)

[Modify the Display of Hit Titles](#)

[Change the Navigation Facets Order](#)



## About Mashup Builder

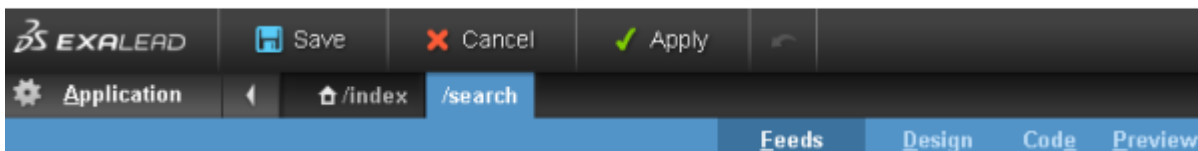
Mashup Builder is a tool that enables you to build search applications.

Each application is composed of pages. For each page, you can configure:

- The data sources to use, called **Feeds**.
- The structure of the page, called **Design**, where you can display feed content in widgets.

The default application that installs with Exalead CloudView contains two pages:

- The **/index** page, which is the home page,
- and the **/search** page, which displays the search results.



## Modify the Display of Hit Titles

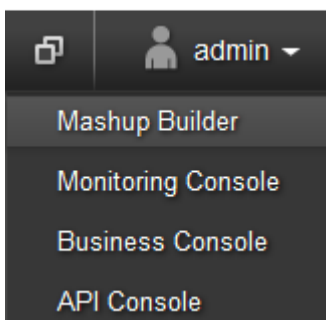
First, let us improve how the hits display in the Mashup UI with dynamic titles.

By default, the title that displays for each hit is based on this value `${entry.metas['title']} | entry.metas['url'] }` where `title` and `url` are default meta names. This expression is defined using the Mashup Expression Language (MEL).

Our `sales` class does not have a `title` meta and the `url` meta displayed by default (as `id=<integer>&`) is not meaningful. We want to display the quantity of each product sold by salesmen and saleswomen.

## Tutorial: Display the Quantity of Each Product Sold by Sales People

1. Open Mashup Builder.
  - Go to `http://<HOSTNAME>:<BASEPORT+1>/mashup-builder`
  - Or from the Administration Console top bar, use the application selector.



2. In Mashup Builder, select the **/search** page, and then the **Design** view.
3. Click the header of the **Result List** widget.
4. On the widget properties panel, define the value of the **Hit title** field as follows:

```
${entry.metas['quantity']} units of ${entry.metas['name']} - sold by  
${entry.metas['firstname']} ${entry.metas['lastname']}
```

To build this expression, select the Exalead CloudView metas from the dynamic list on the left.

5. Remove the thumbnails, as our database records do not have images.
  - a. Select the **Hit thumbnail** tab.
  - b. In **Display thumbnails**, select **none**.
6. Click **Apply**.

### Tutorial: Check the Display of Hit Titles

1. Go to the Mashup UI search page: <http://<HOSTNAME>:<BASEPORT>/mashup-ui/page/search?q=>
2. Refresh the `/search` page to view the results of your changes.

Results › 1-10 of 6270		Sort by Relevance Date Size	
27 units of Fedora hat - sold by Neil Daniels		Download	Preview
category	hat	color	black
description	Classic Fedora hat with pinched crown,	firstname	Neil
lastname	Daniels	name	Fedora hat
quantity	27	store_city	Liverpool
store_country	UK	unit_price	24.99
Source	sales	Data model class	sales
id: id=4&			

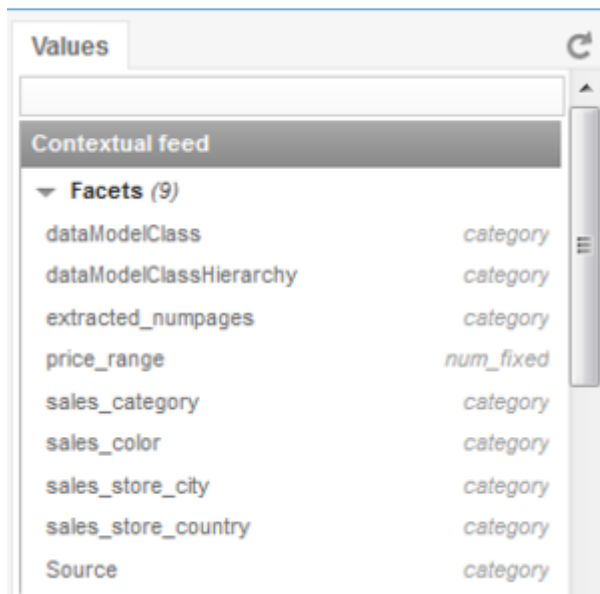
### Change the Navigation Facets Order

Changing the order of navigation facets is one of the most common actions you want to perform when configuring your Mashup UI application. You can specify this sorting for each page of the application.



## Tutorial: Change the Navigation Facets Order in the Refinements Panel

Current order	Order we want
price_range	sales_category
sales_store_city	sales_color
sales_color	price_range
sales_category	sales_store_country
sales_store_country	sales_store_city

1. In Mashup Builder, go to the **/search** page and select the **Design** view.
2. Click the header of the **Standard Facets** widget. This is the widget used by default for the Refinements panel.
3. Specify explicitly the facets to include in the widget:
  - a. In the **Facets** tab, set **Facet root list mode** to **Include**.
  - b. Click inside **Facets list**. A list of available facets displays to the left under **Facets**.



- c. **Select** sales\_category, sales\_color, price\_range, sales\_store\_country, sales\_store\_city

	Facets	General	Advanced
Destination page on click:			
Facet root list mode:	Include		
Facets list:	sales_category, sales_color, price_range, sales_store_country, sales_store_city,		
Aggregation:	count		
Disjunctive facets:	<input type="checkbox"/> 		
Display exclude:	<input type="checkbox"/> 		
Sort facets by:	default		
Sort categories by:	default		
Iteration mode:	all		

4. Click **Apply**.

### [Tutorial: Check the Navigation Facets Display](#)

1. Refresh the /search page in the Mashup UI.

The navigation facets order has changed in the **Refinements** panel.

Refinements	
▼ sales_category	
t-shirt	4047
trousers	2798
shirt	2499
swimwear	2338
shorts	1128
belt	1111
hat	1083
▼ sales_color	
black	6228
white	3016
red	2813
multiple	2338
blue	807
▼ price_range	
\$ 1-21	5778
\$ 21-41	6270
\$ 41-61	1984
\$ 141-161	968
▼ sales_store_country	
USA	3304
France	2977
UK	2817
Germany	2308
Brazil	1383
Japan	1371
China	881
Australia	111
Canada	68
▼ sales_store_city	
Lyon	1551
Paris	1428
Liverpool	1414

# Going Further with CloudView

Up to now we have seen simple use cases to get a general understanding of how to index a data source, search this index, and make some simple modifications to the Mashup UI. The aim of this chapter is to expose advanced capabilities to go further with Exalead CloudView.

When applicable, you will find examples based on the sample database used for the tutorial described in [Creating a Simple Search Application](#). It also contains references to other guides for more detailed information.

[Tools for Analytics](#)

[Expanding User Queries](#)

[Related Terms](#)

[Consolidating Data](#)

[Configuring Results Ordering and Grouping](#)

[Creating Query Alerts](#)

[Securing Document Access](#)

[Customizing CloudView](#)

## Tools for Analytics

You can use Exalead CloudView to build search applications that allow users to drill-down and analyze data.

This section uses the tutorial database to show how you can configure Exalead CloudView for analyzing sales data. Imagine you are the Marketing manager of a clothing store, and you want to see sales figures for countries and cities.

You will learn how to create facets based on aggregations, and display them in the Mashup UI as tables and charts.

**Note:** The examples presented in this section require widgets available in Mashup Builder Premium only.

[Tutorial: Aggregate Facet Values](#)

[Tutorial: Create a Multi-Dimension Facet](#)

[Tutorial: Add a Pivot Table](#)

[Tutorial: Enable Drill-Down Using Hierarchical Facets](#)

## Collapse or Group Search Results

### Tutorial: Aggregate Facet Values

When using the `sales_store_country` facet, we can refine on orders by countries. We now want to aggregate revenues for each country contained in the **country** facet.

**Note:** The Data Table widget used in this section is only available in Mashup Builder Premium.

#### Add an Aggregation to the `sales_store_country` Facet

1. In the Administration Console, select **Search > Search Logics > sl0**.
2. Under the **Facets** section, expand `sales_store_country`.
3. Expand **Aggregations** and specify the following:
  - a. **Id:** Revenue
  - b. **Type:** SUM
  - c. **Expression:** `(sales_unit_price * sales_quantity)`
4. Click **Apply**.

#### Configure the Display of the Facet Aggregation

1. In Mashup Builder, select the `/search` page, and then select the **Design** view.
2. From the **Widgets** panel, expand **Tables** and drag **Data Table** to the page so it is immediately above the **Result List** widget.
3. Click the **Data Table** widget header and on the widget properties panel, specify the following:
  - a. In **Widget title**, enter `Revenue by Country`
  - b. In **Facet name**, select `sales_store_country` (from the **Values** list on the left)

**Note:** If you do not see this facet on this list, click the Reload icon at the top of the Values list.

4. Select the **Column** tab.
  - a. In **Facet column title**, enter `Country`.
  - b. In **Columns**, remove all the columns for **count** and **score**.
5. Still in **Columns**, use the dynamic list on the left to add the following:

Column 1:

- **Value:** Revenue
- **Title:** Revenue (\$)
- **Display percentage:** not selected

Column 2:

- **Value:** Revenue
- **Title:** % of Total
- **Display percentage:** selected

General	Column	Row	Advanced	CSS		
Facet column title: Country						
Total column title: Total						
Columns:		Value	Title	Display percentage		
Revenue		Revenue (\$)		<input type="checkbox"/>		
Revenue		% of Total		<input checked="" type="checkbox"/>		

6. Click **Apply**.

### View Your Changes

1. Go to the Mashup UI, and refresh the search page.

Results 1-10 of 15000			Sort by	Relevance	Date	Size
Revenue by Country						
Country	Revenue (\$)	% of Total				
USA	1896932.26	21.66				
France	1761829.89	20.12				
UK	1585023.87	18.10				
Germany	1378627.97	15.75				
Brazil	780053.6	8.91				
Japan	812465.65	9.28				
China	441949.77	5.05				
Australia	66399.32	0.76				
Canada	32480.7	0.37				
Total	8755763.03	100.00				

2. Click a facet within the table to narrow your search results for one country.




Results › 1-10 of 3304

Sort by Relevance Date Size


Revenue by Country

Country	Revenue (\$)	% of Total
USA <span>×</span>	1896932.26	100.00
Total	1896932.26	100.00

 2 units of Fedora hat - sold by Wyatt Mclean

DownloadPreview

category	hat	color	black
description	Classic Fedora hat with pinched crown,	firstname	Wyatt
lastname	Mclean	name	Fedora hat
quantity	2	store_city	Chicago
store_country	USA	unit_price	24.99
Source	sales	Data model class	sales
id: id=2&			

 7 units of Classy T-Shirt - sold by Dahlia Powell

DownloadPreview

## Tutorial: Create a Multi-Dimension Facet

We now want to determine the number of items sold by city and by product color. City and color will be the two dimensions of a new facet called `units_sold`.

A multi-dimension facet uses the categories generated by two or more facets (any type, virtual or not, but one-dimensional only) to create its categories. Let us use a multi-dimension facet here that is calculated at search-time, meaning it is calculated for the search results only.

By contrast, creating this as an index field would generate a huge number of values since it would be calculated for all documents: `number of field values = number of cities * number of colors`, for all documents in the index.

**Note:** The widget allowing the display of multi-dimension facets is only available in Mashup Builder Premium.

### Add the `units_sold` Multi-Dimension Facet

1. In the Administration Console, select **Search > Search Logics > sl0**.
2. Click **Add facet**, and specify the following:
  - a. In **Name**, enter `units_sold`.
  - b. In **Type**, select **Multi-Dimension**.
3. Specify the following **Dimensions**:

- a. In the first facet, select `sales_color`.
  - b. In the second facet, select `sales_store_city`.
4. Expand **Aggregations**, and specify the following:
  - a. In **Id**, enter `units_sold`.
  - b. In **Type**, select **SUM**.
  - c. In **Expression**, enter `sales_quantity`.
5. Click **Apply**.

### Configure the Display of the Multi-Dimension Facet

1. In Mashup Builder, select the `/search` page, and then select the **Design** view.
2. From the **Widgets** panel, expand **Tables** and drag **2D Data Table** to the page so it is immediately above the **Data Table** widget.
3. Click the **2D Data Table** widget header and specify the following:
  - a. In **Widget Title**, enter `Unit Sold by Country and Color`.
  - b. In **Facet id**, select `units_sold` (from the contextual menu on the left).

**Note:** If you do not see this facet on this list, click the "Reload" icon at the top of the list.

- a. In **Aggregation**, replace the default `count` aggregation by `units_sold`.
  - b. In **Label**, enter `Units Sold`.
5. Click **Apply**.

### View Your Changes

1. Go to the Mashup UI, and refresh the search page.
2. Search for `shirt`

A two-dimensional table now appears in the search results.

Unit Sold by Country and Color					
	white	red	black	blue	Total
Lyon	6370	2473	1828	1948	12619
Liverpool	5184	3784	1726	1311	12005
Paris	4936	2694	1999	1667	11296
Rio de Janeiro	5448	3057	1644	1168	11317
Berlin	4051	2293	1542	1424	9310
London	4239	2549	1044	1076	8908
Chicago	3919	1886	1581	1035	8421
Munich	3499	2296	1446	844	8085
Los Angeles	2616	1300	1170	726	5812
Tokyo	2345	1852	1034	546	5777
Washington D.C	1402	1891	921	738	4952
Kyoto	1487	1034	811	548	3880
San Francisco	1638	665	650	250	3203
Beijing	1301	779	466	624	3170
Hong-Kong	1094	521	538	451	2604
Boston	1060	358	285	209	1912
Shangai	524	282	523	177	1506
New-York	546	280	364	32	1222
Sidney	315	373	64	58	810
Montreal	174	37	59	44	314
Total	52148	30404	19695	14876	117123

3. When you click a value within the table, for example **red** for **Rio de Janeiro**, this narrows the search results on both dimensions.

Unit Sold by Country and Color		
	red x	Total
Rio de Janeiro x	3057 x	3057
Total	3057	3057

## Tutorial: Add a Pivot Table

We want to configure the Mashup UI to display a Pivot Table widget on which the user can refine the search results.

For our example, we want to see the sales quantity for each cloth article by country .

**Note:** This widget is only available in Mashup Builder Premium.

## Configure the Pivot Table Widget

1. In Mashup Builder, go to the **/search** page and select **Design**.
2. From the **Widgets** panel, select **Table > Pivot Table**.
3. Drag the **Pivot Table** on to the **/search** page so it is just below the **Navigation Header** widget.
4. Click the **Pivot Table** widget header.
5. In the **Search API** tab, for **Search Logic**, select `s10`.
6. To configure the table header, go to **Columns**:
  - a. In **Facet name**, select `sales_category`
  - b. In **Sort function**, select `alphanum`
  - c. In **Legend**, enter `Clothes`.
  - d. Select **Enable column drill-down**.

The screenshot shows the configuration interface for the Pivot Table widget, specifically the **Columns** tab. The interface has a blue header bar with tabs: **General**, **Search API**, **Columns** (selected), and **Rows** (with a warning icon and the number 2). Below the header, there are three main sections: **Columns:**, **Facet name**, **Sort function**, and **Legend**. The **Columns:** section contains a text input field with the value `sales_category`. The **Sort function** section contains a dropdown menu with the value `alphanum`. The **Legend** section contains a text input field with the value `Clothes`. At the bottom, there is a checkbox labeled **Enable column drill-down:** which is checked.

7. To configure row display by countries then by cities, go to **Rows** and for:
  - a. **Legend**, enter `Country/City`.
  - b. For the first row: in **Facet name**, select `sales_store_country`; in **Sort function**, select `alphanum`
  - c. For the second row: in **Facet name**, select `sales_store_city`; in **Sort function**, select `alphanum`
  - d. For **No. facet levels shown by default**, enter `1` to display by default the first facet level only (`sales_store_country`).

The screenshot shows the configuration interface for the Pivot Table widget, specifically the **Rows** tab. The interface has a blue header bar with tabs: **General**, **Search API**, **Columns**, and **Rows** (selected). Below the header, there are three main sections: **Legend:**, **Rows:**, and **No. facet levels shown by default:**. The **Legend:** section contains a text input field with the value `Country/City`. The **Rows:** section contains a table with two rows. The first row has **Facet name** `sales_store_country` and **Sort function** `alphanum`. The second row has **Facet name** `sales_store_city` and **Sort function** `alphanum`. The **No. facet levels shown by default:** section contains a text input field with the value `1`.

8. To configure the sales quantity display, go to **Aggregations**:
  - a. In **Type**, select `COUNT`
  - b. In **Expression**, select `sales_quantity`

- c. In **Max decimals**, do not select anything. This field applies to float values (an empty value shows all available decimals).
- d. In **Legend**, enter `Sales quantity`

Aggregations: Type **COUNT** Expression `sales_quantity` Max No. decimals Max No. decimals Legend `Sales quantity` + X ↑

9. Click **Apply**.

### View Your Changes

1. Go to the Mashup UI.
2. Refresh the screen.

In the Mashup UI, you can now see the **Pivot Table** widget on the search page.

*Pivot table showing clothes by country/city, with a collapsed view on the country facet level*

Results 1-10 of 15000								Sort by		Relevance	Date	Size
Clothes	belt	hat	shirt	shorts	swimwear	t-shirt	trousers					
Country/City	Sales quantity	Sales quantity	Sales quantity	Sales quantity	Sales quantity	Sales quantity	Sales quantity					
▶ Australia	7	11	24	10	14	23	22					
▶ Brazil	112	102	223	98	219	377	252					
▶ Canada	6	4	7	7	10	19	15					
▶ China	58	66	142	60	130	249	156					
▶ France	248	207	500	220	463	784	555					
▶ Germany	158	154	383	185	370	604	454					
▶ Japan	102	87	221	120	215	355	271					
▶ UK	184	193	456	172	407	729	476					
▶ USA	236	259	543	254	508	907	597					

3. You can expand the view for a specific country, for example **China**, and see values for its cities (**Beijing, Hong Kong, Shanghai**).

**Note:** You can also click any value of the pivot table to refine results on the whole page.

### Tutorial: Enable Drill-Down Using Hierarchical Facets

In the previous section, we added a widget that displayed revenue by country on the Mashup UI. The next logical step is to add the ability to drill-down on country so we can view revenue by city as well.

We need to create a hierarchical facet that takes the form `Top/ClassProperties/Country/City`.

- `Top/ClassProperties` is generated automatically when you add a new property to the Data Model as a facet.
- We can also generate `Country/City` by concatenating the `store_country` and `store_city` metas.

Since concatenation is a more advanced type of document analysis, we need to configure a new **document processor**.

**Note:** The Pie Chart widget used in this section is only available in Mashup Builder Premium.

### Configure the Concatenate Values Document Processor

Document processors analyze document metas, then output them as contexts, which you can then define as index fields and as facets in the Data Model.

Up until now, when you added a new property to the Data Model, there were processors already in place to perform the required document analysis.

To be able to drill-down from country to city, however, you must create a facet that emulates a hierarchy. Facets created in the Data Model have this value in the Index: `Top/ClassProperties/<FACET NAME>`

To make a hierarchical facet, you need to add another level, to denote the child facet, separated by a "/" from its parent: `Top/ClassProperties/<PARENT FACET NAME>/<CHILD FACET NAME>`

1. In the Administration Console, go to **Index > Data Processing > ap0 > Document Processors**.
2. Go to the **Processor types** list and expand **Text operations**.
3. Drag the **Concatenate Values** document processor to the end of the **Current processors** list.
4. Expand the processor and complete its fields:
  - a. **Output to:** `sales_country_city`
  - b. **Join:** `/`
  - c. **Input from:** `sales_store_country`
  - d. **Input from:** `sales_store_city`
5. Click **Save**.

We now need to add a new property to the `sales` class that will use the `sales_country_city` output meta.

## Add a New Data Model Property Using the Output Meta

1. Go to **Index > Data Model**.
2. Select the `sales` class.
3. Click **Add property**.
  - a. **Name:** type `country_city`
  - b. **Data Type:** select **Alphanum**
  - c. **Semantic Type:** select **metadata**
  - d. **Field Type:** select **Facet only**
  - e. Click **Accept**.
4. Click **Apply**.
5. Go to the **Home** page, **Clear** the index.
6. Once cleared, click **Scan** for the `sales` connector.

## Configure the Display of the Hierarchical Facet

1. In Mashup Builder, go to the `/search` page and select **Design**.
2. From the **Widgets** panel, select **Visualizations > Charts**.
3. Drag the **Pie Chart** to the `/search` page so it is just below the **Navigation Header** widget.
4. Click the widget header and on the widget properties panel, specify the following:
  - a. In **Widget Title**, enter `Sales by Country, then City`.
  - b. In **Facet**, select `sales_country_city`.

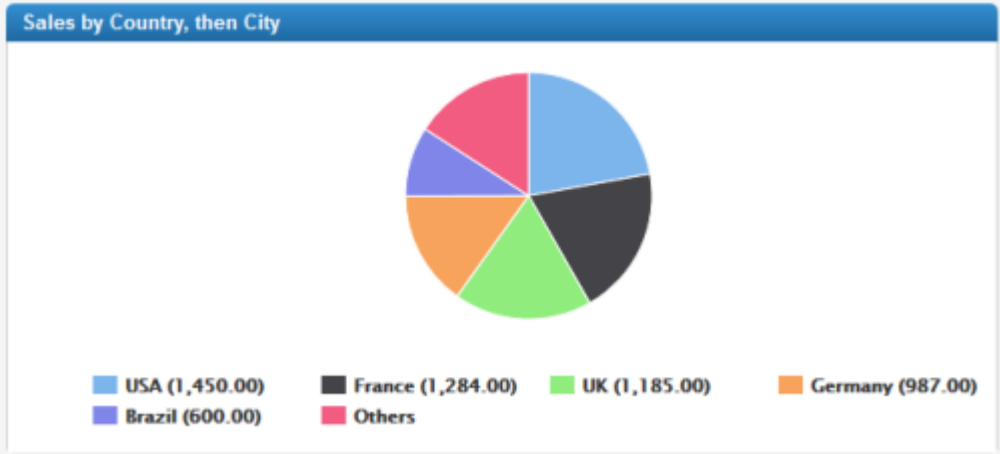
### Note:

If you do not see this facet on this list, click the **Reload** icon at the top of the list.

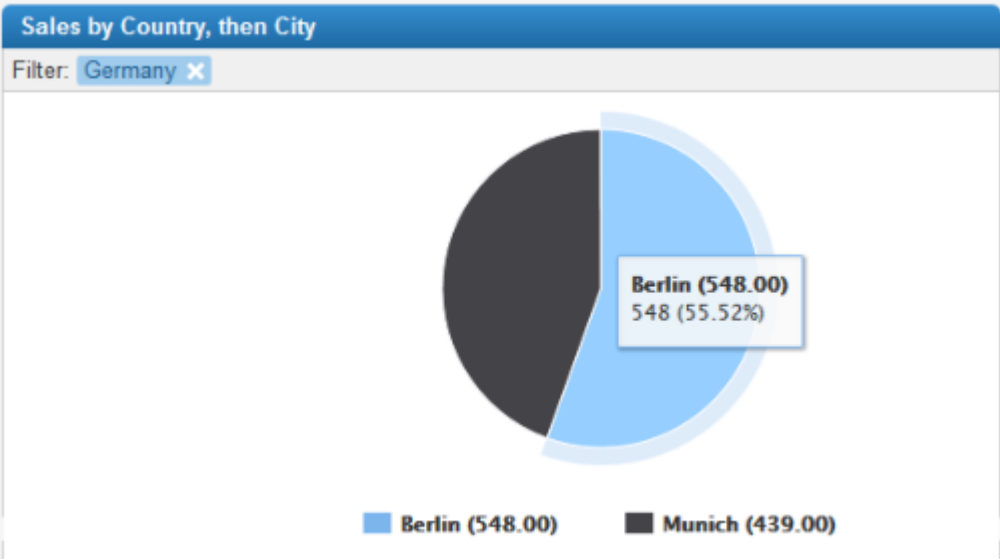
5. Click **Apply**.

## View Your Changes

1. Go to the Mashup UI.
2. Refresh the screen.
3. When you search for `shirt` in the Mashup UI, a pie chart now appears in the search results.



When you click a slice, the chart drills-down to the cities belonging to the parent country.



Collapse or Group Search Results

Collapsing or grouping search results means keeping similar results together so they display in a concise, readable way.

There are typically two use cases where you want to collapse search results.

Use case	Description
See a concise list of results, often based on multiple criteria	<div>You do not want to explore, you just want to see specific hits.</div> <div>Examples:</div> <div><ul style="list-style-type: none"><li>For web search, you want to keep only one result per site.</li></ul></div>



Use case	Description
	<ul style="list-style-type: none"> <li>You want a list of every Apple phone that had more than 1000 units sold in the past week, worldwide, regardless of store or country.</li> </ul> <p>To do this, use grouping, as explained in this section.</p>
See the overall distribution of search results in a dashboard	<p>Here you want to explore the results by drilling down on certain areas.</p> <p>For example, you want to know how different Apple products are selling in different countries.</p> <p>For more information, see "Collapsing/ Grouping Search Results" in the <i>Exalead CloudView Configuration Guide</i>.</p>

## Expanding User Queries

Query expansion essentially adds additional search terms to the user's original query. For example, if you enable phonetic query expansion, the query `exaleed` will be expanded to `"exaleed" OR "exalead"`.

### About Query Expansion Features

[Tutorial: Add Suggest on the Name Field](#)

### About Query Expansion Features

Query expansion features are essential to help users entering their queries.

For more information about query expansion features, see "Configuring Query Expansion" in the *Exalead CloudView Configuration Guide*.

### Spell check

You can enable your search logic to include spell-check, which suggests alternate words or expressions to replace the original user query. Spell-check suggestions are determined by a score. The score is calculated based on:

- First, the Damerau-Levenshtein transformation distance from the original word
- Second, the frequency of terms in your corpus.

The transformation distance is the number of changes (replace a letter, add a letter, transpose a letter, delete a letter) required to transform word A to word B. Each change represents a distance. The greater the transformation distance, the larger the difference between two words.

## Synonyms

The Synonym expansion module adds alternative forms to user queries. For example, if the text prefix handler uses the synonyms module, the query: "db architect" expands to "db architect" OR "data base architect" OR "database architect".

Unlike the other query expansion modules, you must first compile your own synonym dictionary, also known as a resource file, that defines the possible synonyms for a particular expression.

## Phonetization

Phonetization is the ability to search for alternative forms that sound like the original query word, for example (soundslike: exaleed).

## Approximation

For an example of approximation use, see [Use Semantic Analysis to Correct Queries](#).

## Wildcard Search

Wildcard search is a pattern-matching feature enabled by default. It allows you to find documents that include "test", "tests" and "tesselation" when searching on "tes\*". Additional configuration is available to fine-tune your results.

If **Search > Search Logics > Query Expansion > Wildcard Search** is enabled, wildcard search will be expanded using a dictionary generated from the corpus.

## Search Suggest

The goal of search suggestion is to auto-complete the user's query by providing relevant suggestions for what the user wants to search. In other words, it shows some of the terms associated with the beginning of the user search query.

Search suggest relies on precomputed dictionaries to offer efficient matching (millisecond-range, thousands of queries per second). It can be based on:

- Exalead CloudView index content – fetching the values of an index field or a category facet.
- Previously-performed queries.
- Custom dictionaries provided by the administrator.

Suggest dictionaries are recomputed periodically.

A suggest dictionary contains suggest entries. These entries are the suggestions to be made to the user. Each entry has a given score. The number of possible matches for a given input string is a fixed parameter when building a suggest. For each input string, only the N best matches can be returned.

## Tutorial: Add Suggest on the Name Field

In this tutorial, we want to add suggest based on the `name` field from our sample sales database, so that when a user starts entering `sl`, we automatically suggest `sleeveless shirt`, `sleeve shirt`, and `slim shorts`.

For that, we need to build a suggest dictionary based on the content of the `sales_name` index field (see the **Data Model > Advanced Schema** tab for a listing of all index fields), and set up the Mashup UI to use this suggest data.

For more information, see "Adding Search Suggestions" in the *Exalead CloudView Configuration Guide*.

### Build a Suggest Dictionary

1. In the Administration Console, select **Search > Suggest**.
2. Click **Add suggest**:
  - a. For **Name**, type, for example, `name_suggest`.
  - b. For **Type**, select **Index field suggest**.
  - c. Click **Accept**.
3. In the configuration options for this new suggest, click in the **Index field** and select `sales_name` from the list.
4. We want to find matches not only for the first word of the name value, but for each word:
  - a. Expand **Build options**.
  - b. Select **Subexpr matching**.
5. Click **Apply**.

Now that you have declared the suggest, you must precompile data for efficient lookup.

6. Under **Suggest status**, click **Build now**.

Now, we need to add the suggest to the Mashup UI.

### Add the Suggest to the Mashup UI

1. In Mashup Builder, go to a page using a search form widget. For example, the `/index` page, which uses the **Standard Search Form** widget.
2. Click the widget header to display its properties panel.

3. On the **Suggest** tab, complete the following:
  - a. Select **Enable suggest**.
  - b. For **Suggest Name**, click inside the field.
  - c. From the dynamic list that displays on the left, select the suggest service.

**Note:**

If you do not see the suggest you just created, refresh the list.

4. Repeat these steps for the **/search** page.
5. Click **Apply**.

### Test the Suggest in the Mashup UI

1. Go to the Mashup UI, and refresh the search page.
2. Start entering `sl` in the **Search** box.

The search box suggests all the available values that start with `sl`.

**Note:** If you do not see the following suggestions, empty your browser cache.

sl	Search
Smart slim shorts	
Sleeveless shirt	
Sleeve shirt	
Classic sleeve Shirt	

## Related Terms

You can use related terms to offer users related query terms, that might be relevant depending on their original queries.

For more details, see "Adding Related Terms" in the *Exalead CloudView Configuration Guide*.

## Consolidating Data

Consolidation is very helpful for incremental indexing, to take updates as they come instead of rebuilding the entire index when an object changes.

The incremental update is a complex task as it requires calculating the impact of any changes and building complete documents according to projection rules.

**Note:** The Exalead CloudView Consolidation Server is not limited to one data source. It can work across several data sources, which allows building documents based on objects coming from different sources. This avoids having an ETL or an equivalent tool to perform cross-source joins and aggregations.

For more information, see the *Exalead CloudView Consolidation Server Guide*.

## Configuring Results Ordering and Grouping

The order in which hits display in the search results is controlled by the ranking and sorting you define for a search logic.

There are two phases in ranking and sorting: calculating the ranking elements, then defining sorting and grouping using these ranking elements.

For more information, see "Collapsing/ Grouping Search Results" in the *Exalead CloudView Configuration Guide*.

## Creating Query Alerts

You can set up query alerting on your Mashup Builder application. This enables end users to save queries so they are alerted whenever new or modified documents that match the query are added to the index.

These alerts can either be:

- Scheduled - the alert runs at a fixed time, at regular intervals.
- Real time - the alert runs as soon as a new matching document is indexed.

For more information, see "Setting up Query Alerting" the Exalead CloudView Business Console User's Guide.

## Securing Document Access

Security in Exalead CloudView has different levels:

- Document-level security ensures that documents which users are authorized to view are displayed as search results from the Exalead CloudView index.
- Application-level security ensures a restricted access to features in the Administration Console and Business Console.

This section focuses on applying document-level security.

For more information, see "Managing User Access" in the *Exalead CloudView Administration Guide* .

## [How Document Security Works](#)

### [Tutorial: Implement Document Security](#)

## How Document Security Works

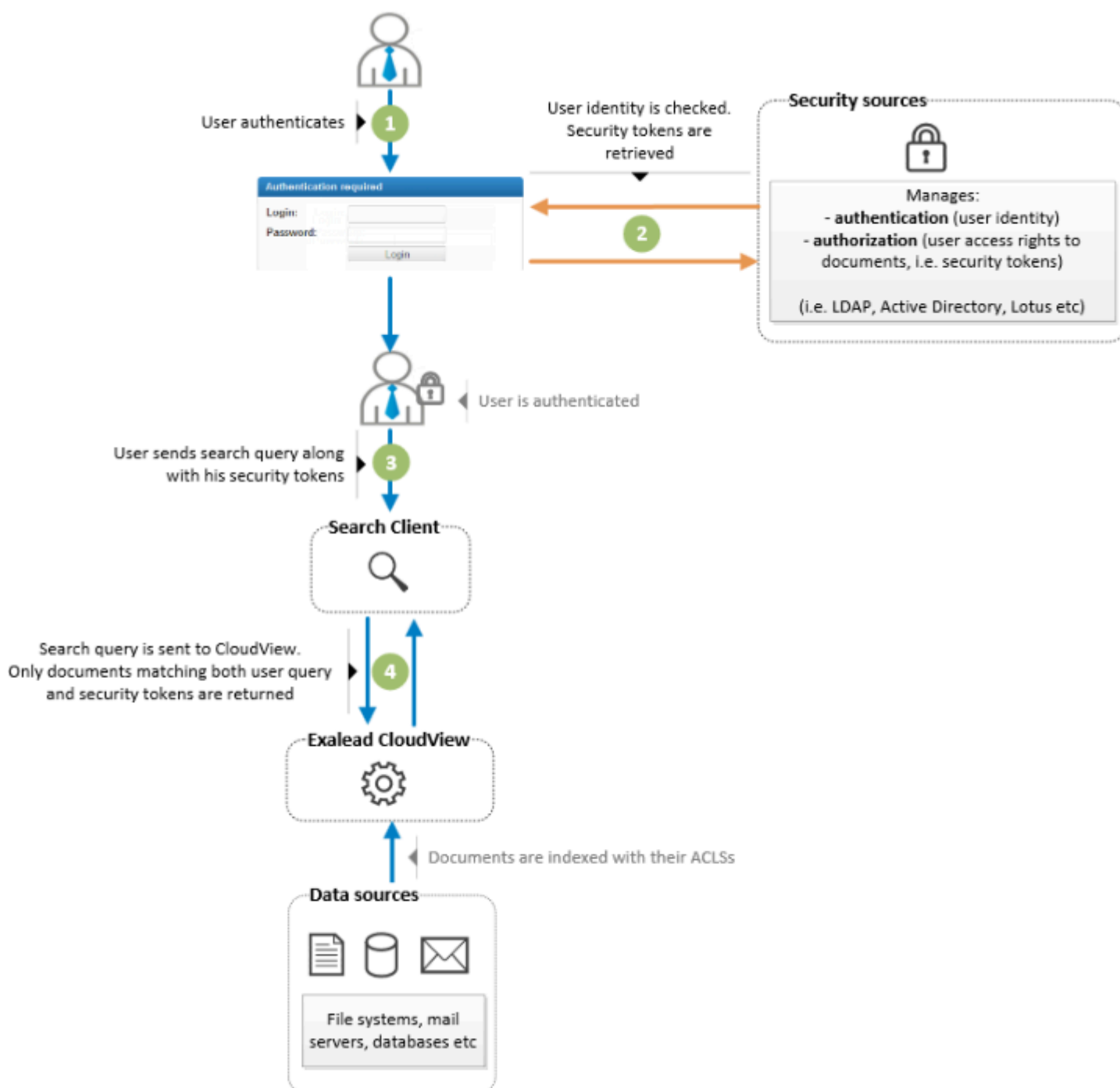
Document security is implemented by indexing a document's Access Control List (ACL) and generating security tokens when the user authenticates.

If your network already has a security policy in place, you may want to reuse it for authentication in Exalead CloudView.

**Note:** In Exalead CloudView, password, and login management is centralized and changes are automatically propagated.

Configure a security source to authenticate users and authorize their document access.

The document security workflow is described below.



Step	Description
1	The user logs in to the Exalead CloudView search application.
2	Exalead CloudView queries a security source for: <ul style="list-style-type: none"> <li>Authentication: to verify the user's login and password</li> <li>Authorization: to verify the security tokens for the user and the group to which the user belongs</li> </ul>
3	The user enters his search query.
4	The user's query and security tokens are sent to the index.

Step	Description
	The index only fetches documents that match both the user's query and the user's security tokens.

## Tutorial: Implement Document Security

As already stated, Exalead CloudView queries a security source for:

- Authentication: to verify the user's login and password
- Authorization: to verify the security tokens for the user and the group to which the user belongs

This section will show how to configure user authentication only.

### Create the Security Source

1. In the Administration Console, go to **Search > Security Sources** and click **Add security source**.
  - a. For **Name**, type `MySecurity`.
  - b. For **Type**, select **Simple Security**.
2. For this security source, under **Config > Users**:
  - a. Click **Add item** to add a new user.
  - b. Specify user login information: **Login**: `myuser`, **Password**: `myuser`, **Display name** (the name that will be displayed when the user is authenticated): `myuser`
3. Click **Apply**.
4. Under **Test user authentication**:
  - a. Click **Test**.
  - b. Enter the user **Login** and **Password**, and click **Test**.

If successful, your user ID, display name, and full list of security tokens (if any) displays.

If not, check your parameters and contact your system administrator.



## Enable Security in the Search Application

We now have a security source which can be queried to check logins. We now need to enable security in the UI, and tell it to use the security source we just created.

1. In Mashup Builder, select **Application** from the top left menu bar.
2. Under **General > Security**, click **Add a security provider**.
3. In the **Add security provider** dialog box, select **CloudView Security Provider** and click **OK**.

### Note:

A **login** page is created in your application.

4. In the **Source** field, select the security source you created in the previous procedure, for example `MySecurity`.
5. In the **Mashup pages** section, select the pages of your application on which you want to enable security. For example, the **index** and **search** pages.

### CloudView Security Provider

 Uses standard CloudView security sources.

API Config:	sapi0
API Endpoints:	
Command:	security
Source:	MySecurity
Authenticate to:	FIRST 

[Remove security provider](#)

### Mashup pages

 Enables the security on the following Mashup pages:

index: ☒ search: ☒

6. For each page on which security is enabled, also add the **Logout** widget:
  - a. Display the **search** page.
  - b. In **Widgets**, search for 'Logout'.
  - c. Drag the **Logout** widget to the page.
  - d. Repeat these steps for the **index** page.
7. Click **Apply** to save the configuration changes.

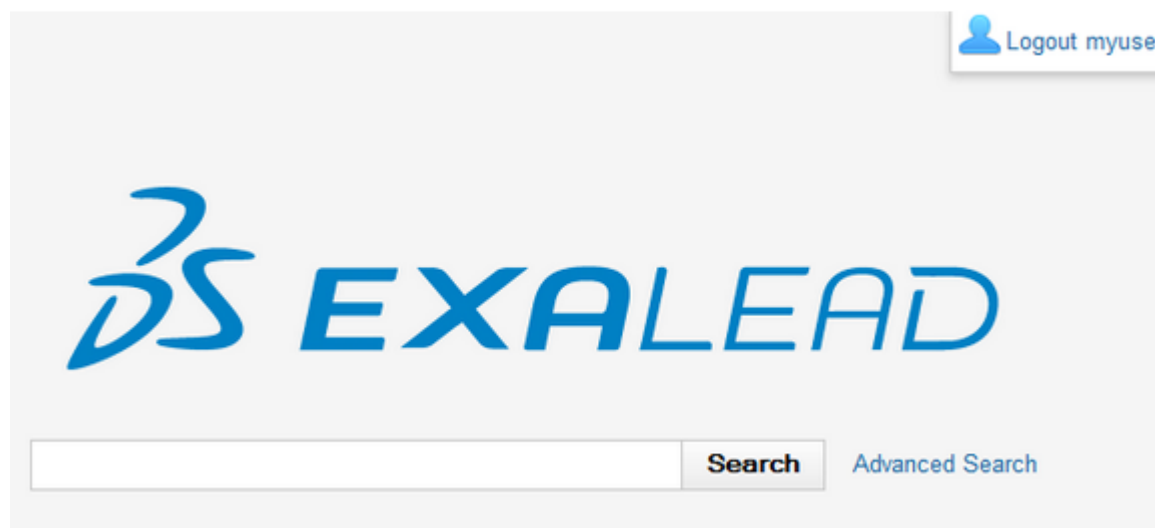
## Test Authentication

1. Go to the Mashup UI.

You are prompted to log in.

2. Log in with the login/password defined previously (`myuser`).

Once logged in, you can see the logout widget on the page (if you added the Logout widget).



## Customizing CloudView

If you have programming skills, Exalead CloudView provides several APIs to allow integration with third-party applications.

- The Push API (PAPI) is the public API that allows data from any source to be indexed by Exalead CloudView. It supports all operations required to develop new connectors, both managed and unmanaged.

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

- The Mashup API is the API which retrieves the contents of data sources to make them accessible to the data feeds.

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

For more information, see "Using the Mashup API" in the *Exalead CloudView Mashup Programmer's Guide*.

- The Search API is the public API for developing third-party search applications. It is the entry point for performing searches on Exalead CloudView.

For more information, see "Appendix - Search API Parameters" in the Exalead CloudView Configuration Guide.

- The Management API (MAMI) is the public API used to configure and manage the Exalead CloudView processes.

For more information, see "The Management API (MAMIs)" in the *Exalead CloudView Programmer's Guide*.