Cloudera Data Management

Cloudera, Inc.
1001 Page Mill Road, Bldg 3
Palo Alto, CA 94304
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Navigator 2.8.x
Date: December 1, 2016

Table of Contents

Cloudera Navigator Lineage Diagrams...58

About Cloudera Data Management

This guide describes how to perform data management using Cloudera Navigator. Data management activities include auditing access to data residing in HDFS and Hive metastores, reviewing and updating metadata, and discovering the lineage of data objects.

> **Important:** This feature is available only with a Cloudera Enterprise license. It is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see Managing Licenses.

Cloudera Navigator is a fully integrated data-management and security system for the Hadoop platform. Cloudera Navigator enables you to work effectively with data at scale and helps various stakeholders answer the following questions:

- Compliance groups

    - Who accessed the data, and what did they do with it?
    - Are we prepared for an audit?
    - Is our sensitive data protected?

- Hadoop administrators and DBAs

    - How can we boost productivity and cluster performance?
    - How is data being used?
    - How can data be optimized for future workloads?

- Data stewards and curators

    - How can data assets be managed and organized?
    - What is the lifecycle of the data?

- Data scientists and Business Intelligence users

    - Where is the most important data?
    - Is this data trustworthy?
    - What is the relationship between data sets?

Cloudera Navigator provides the following components to help you answer these questions and meet data-management and security requirements.

- **Data Management** - Provides visibility into and control over the data in Hadoop datastores, and the computations performed on that data. Hadoop administrators, data stewards, and data scientists can use Cloudera Navigator to:

    - Audit data access and verify access privileges - The goal of auditing is to capture a complete and immutable record of all activity within a system. Cloudera Navigator auditing adds secure, real-time audit components to key data and access frameworks. Compliance groups can use Cloudera Navigator to configure, collect, and view audit events that show who accessed data, and how.
    - Search metadata and visualize lineage - Cloudera Navigator metadata management allows DBAs, data stewards, business analysts, and data scientists to define, search for, amend the properties of, and tag data entities and view relationships between datasets.
    - Policies - Data stewards can use Cloudera Navigator policies to define automated actions, based on data access or on a schedule, to add metadata, create alerts, and move or purge data.
    - Analytics - Hadoop administrators can use Cloudera Navigator analytics to examine data usage patterns and create policies based on those patterns.

- **Data Encryption** - Data encryption and key management provide a critical layer of protection against potential threats by malicious actors on the network or in the datacenter. Encryption and key management are also

requirements for meeting key compliance initiatives and ensuring the integrity of your enterprise data. The following Cloudera Navigator components enable compliance groups to manage encryption:

– Cloudera Navigator Encrypt transparently encrypts and secures data at rest without requiring changes to your applications and ensures there is minimal performance lag in the encryption or decryption process.
– Cloudera Navigator Key Trustee Server is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys and other security artifacts.
– Cloudera Navigator Key HSM allows Cloudera Navigator Key Trustee Server to seamlessly integrate with a hardware security module (HSM).

You can install Cloudera Navigator data management and data encryption components independently.

Cloudera Navigator Metadata Architecture

Cloudera Navigator metadata provides data discovery and data lineage functions. The following figure depicts the Cloudera Navigator metadata architecture.
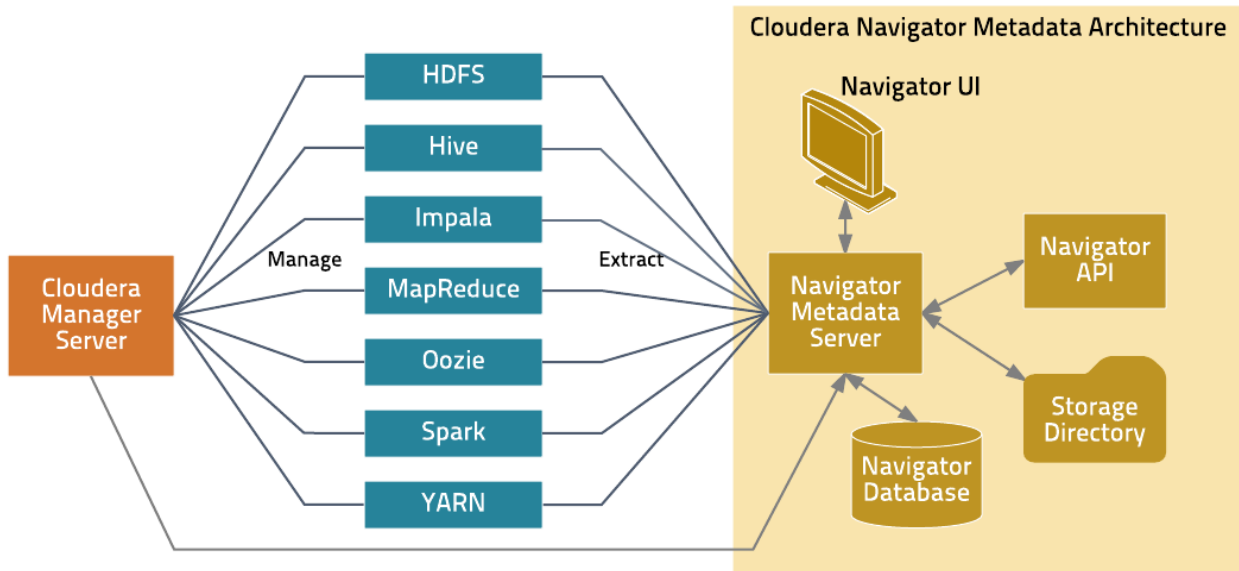
The Navigator Metadata Server performs the following functions:

- Obtains connection information about CDH services from the Cloudera Manager Server
- At periodic intervals, extracts metadata for the entities managed by those services
- Manages and applies metadata extraction policies during metadata extraction
- Indexes and stores entity metadata
- Manages authorization data for Cloudera Navigator users
- Manages audit report metadata
- Generates metadata and audit analytics
- Implements the Navigator UI and API

The Navigator database stores policies, user authorization and audit report metadata, and analytic data. The storage directory stores the extraction state and extracted metadata.

The Cloudera Navigator Metadata Server manages metadata about the entities in a CDH cluster and relations between the entities. The metadata schema defines the types of metadata that are available for each entity type it supports.

For example, the following figure shows the entity details of a file entity:

Cloudera Navigator Metadata Architecture

Three classes of metadata are defined for all entities:

- **Technical Metadata** - Metadata defined when entities are extracted. Such metadata includes:

  - Name of an entity
  - Service that manages or uses the entity
  - Type
  - Path to the entity
  - Date and time of creation
  - Access permissions
  - Modification, size, owner, purpose, and relations—parent-child, data flow, and instance of—between entities

  You cannot modify technical metadata.

- **Custom Metadata** - Descriptions, key-value pairs, and tags that can be added to entities. You can add and modify custom metadata *before and after* entities are extracted.
- **Managed Metadata** - Key-value pairs that can be added to entities. Managed metadata key-value pairs are similar to custom metadata key-value pairs, but can also define the keys within a namespace and enforce conformance to value constraints (for example, require the value to be a date). You can add and modify managed metadata *after* entities are extracted.

In addition, for Hive entities, Cloudera Navigator supports extended attributes, which are added by Hive clients before entities are extracted.

Metadata Extraction and Indexing

Metadata Extraction

The Navigator Metadata Server extracts metadata for the following resource types.

Table 1: Resource Metadata Extraction

Resource Type	Metadata Extracted
HDFS	HDFS metadata at the next scheduled extraction run after an HDFS checkpoint. If high availability is enabled, metadata is extracted as soon as it is written to the JournalNodes.
Hive	Database, table, and query metadata from Hive lineage logs. See Managing Hive and Impala Lineage Properties.
Impala	Database, table, and query metadata from the Impala Daemon lineage logs. See Managing Hive and Impala Lineage Properties.
MapReduce	Job metadata from the JobTracker. The default setting in Cloudera Manager retains a maximum of five jobs; if you run more than five jobs between Navigator extractions, the Navigator Metadata Server extracts the five most recent jobs.
Oozie	Oozie workflows from the Oozie Server.
Pig	Pig script runs from the JobTracker or Job History Server.
Spark	Spark job metadata from YARN logs. (Unsupported and disabled by default. To enable, see Enabling Spark Metadata Extraction.)
Sqoop 1	Database and table metadata from Hive lineage logs; job runs from the JobTracker or Job History Server.
YARN	Job metadata from the ResourceManager.

> **Important:** Tables created by Impala queries and Sqoop jobs are represented as Hive entities.

If an entity is created at time t0 in the system, that entity is extracted and linked in Navigator after the extraction poll period (10 minutes by default) plus a service-specific interval, as follows:

- **HDFS**: t0 + (extraction poll period) + (HDFS checkpoint interval (1 hour by default))
- **HDFS + HA**: t0 + (extraction poll period)
- **Hive**: t0 + (extraction poll period) + (Hive maximum wait time (60 minutes by default))
- **Impala**: t0 + (extraction poll period)

Metadata Indexing

After metadata is extracted, it is indexed and made available for searching by an embedded Solr engine. The Solr schema indexes two types of metadata: entity properties and relationships between entities.

You can search entity metadata using the Navigator UI and API. Relationship metadata is implicitly visible in lineage diagrams and explicitly available by downloading the lineage using the Cloudera Navigator Data Management API.

Metadata Search Syntax and Properties

In Cloudera Navigator, metadata search is implemented by an embedded Solr engine that supports the syntax described in `LuceneQParserPlugin`.

Search Syntax

You construct search strings by specifying the value of a [default property](#) and four types of key-value pairs, using the indicated syntax:

- **Technical metadata key-value pairs** - *key*:*value*
  - *key* is one of the properties listed in [Search Properties](#) on page 11.
  - *value* is a single value or range of values specified as [*value1* TO *value2*]. In a value, * is a wildcard. In property values, you must escape special characters :, -, /, and * with the backslash character (\), or enclose the property value in quotes.

  Technical metadata key-value pairs are read-only and cannot be modified.

- **Custom metadata key-value pairs** - `up_key:value`
  - *key* is a user-defined property.
  - *value* is a single value or range of values specified as [*value1* TO *value2*]. In a value, * is a wildcard. In property values, you must escape special characters :, -, /, and * with the backslash character (\), or enclose the property value in quotes.

  Custom metadata key-value pairs can be modified.

- **Hive extended attribute key-value pairs** - `tp_key:value`
  - *key* is an extended attribute set on a Hive entity. The syntax of the attribute is specific to Hive.
  - *value* is a single value supported by the entity type.

  Hive extended attribute key-value pairs are read-only and cannot be modified.

- **Managed metadata key-value pairs** - *namespace*.*key*:*value*
  - *namespace* is the namespace containing the property. See [Defining Managed Metadata](#) on page 19.
  - *key* is the name of a managed metadata property.
  - *value* is a single value, a range of values specified as [*value1* TO *value2*], or a set of values separated by spaces. In a value, * is a wildcard. In property values, you must escape special characters :, -, /, and * with the backslash character (\), or enclose the property value in quotes.

  Only the values of managed metadata key-value pairs can be modified.

Constructing Compound Search Strings

To construct compound search strings, you can join multiple property-value pairs using the [Lucene Query Parser Boolean operators](#):

- , +, -
- OR, AND, NOT

In both syntaxes, you use () to group multiple clauses into a single field and to form subqueries. When you [filter results](#) in the Navigator Metadata UI, the constructed search strings use the , +, - syntax.

Example Search Strings

- Entities in the path /user/hive that have not been deleted - +("/user/hive") +(-deleted:true)
- Descriptions that start with the string "Banking" - description:Banking*
- Entities of type MapReduce or entities of type Hive - sourceType:mapreduce sourceType:hive or sourceType:mapreduce OR sourceType:hive
- Entities of type HDFS with size equal to or greater than 1024 MiB or entities of type Impala - (+sourceType:hdfs +size:[1073741824 TO *]) sourceType:impala

- Directories owned by `hdfs` in the path `/user/hdfs/input` - `+owner:hdfs +type:directory +fileSystemPath:"/user/hdfs/input"` or `owner:hdfs AND type:directory AND fileSystemPath:"/user/hdfs/input"`
- Job started between 20:00 to 21:00 UTC - `started:[2013-10-21T20:00:00.000Z TO 2013-10-21T21:00:00.000Z]`
- Custom key-value - `project-customer1` - `up_project:customer1`
- Technical key-value - In Hive, specify table properties like this:

```
ALTER TABLE table_name SET TBLPROPERTIES ('key1'='value1');
```

To search for this property, specify `tp_key1:value1`.
- Managed key-value with multivalued property - `MailAnnotation.emailTo:"dana@example.com" MailAnnotation.emailTo:"lee@example.com"`

> **Note:** When viewing MapReduce jobs in the Cloudera Manager Activities page, the string that appears in a job Name column equates to the `originalName` property. To specify a MapReduce job name in a search, use the string `(sourceType:mapreduce)` and `(originalName:jobName)`, where *jobName* is the value in the job Name column.

Search Properties

The following reference describes search schema properties.

Default Properties

The following properties can be searched by specifying a property value: `type`, `fileSystemPath`, `inputs`, `jobId`, `mapper`, `mimeType`, `name`, `originalName`, `outputs`, `owner`, `principal`, `reducer`, and `tags`.

Common Properties

Name	Type	Description
description	text	Description of the entity.
group	caseInsensitiveText	The group to which the owner of the entity belongs.
name	ngramedText	The overridden name of the entity. If the name has not been overridden, this value is empty. Names cannot contain spaces.
operationType	ngramedText	The type of an operation: • Pig - SCRIPT • Sqoop - Table Export, Query Import
originalName	ngramedText	The name of the entity when it was extracted.
originalDescription	text	The description of the entity when it was extracted.
owner	caseInsensitiveText	The owner of the entity.
principal	caseInsensitiveText	For entities with type `OPERATION_EXECUTION`, the initiator of the entity.
properties	string	A set of key-value pairs that describe the entity.
tags	ngramedText	A set of tags that describe the entity.
type	tokenizedCaseInsensitiveText	The type of the entity. The available types depend on the entity's source type: • `hdfs` - DIRECTORY, FILE, DATASET, FIELD

Name	Type	Description
		• `hive` - DATABASE, TABLE, FIELD, OPERATION, OPERATION_EXECUTION, SUB_OPERATION, PARTITION, RESOURCE, VIEW • `impala` - OPERATION, OPERATION_EXECUTION, SUB_OPERATION • `mapreduce` - OPERATION, OPERATION_EXECUTION • `oozie` - OPERATION, OPERATION_EXECUTION • `pig` - OPERATION, OPERATION_EXECUTION • `spark` - OPERATION, OPERATION_EXECUTION • `sqoop` - OPERATION, OPERATION_EXECUTION, SUB_OPERATION • `yarn` - OPERATION, OPERATION_EXECUTION, SUB_OPERATION
`userEntity`	Boolean	Indicates whether an entity was added using the Cloudera Navigator SDK.
Query		
`queryText`	string	The text of a Hive, Impala, or Sqoop query.
Source		
`clusterName`	string	The name of the cluster in which the source is managed.
`sourceId`	string	The ID of the source type.
`sourceType`	caseInsensitiveText	The source type of the entity: `hdfs`, `hive`, `impala`, `mapreduce`, `oozie`, `pig`, `spark`, `sqoop`, or `yarn`.
`sourceUrl`	string	The URL of web application for a resource.
Timestamps		
The available timestamp fields vary by the source type: • `hdfs` - created, lastAccessed, lastModified • `hive` - created, lastModified • `impala`, `mapreduce`, `pig`, `spark`, `sqoop`, and `yarn` - started, ended	date	Timestamps in the Solr Date Format. For example: • `lastAccessed:[* TO NOW]` • `created:[1976-03-06T23:59:59.999Z TO *]` • `started:[1995-12-31T23:59:59.999Z TO 2007-03-06T00:00:00Z]` • `ended:[NOW-1YEAR/DAY TO NOW/DAY+1DAY]` • `created:[1976-03-06T23:59:59.999Z TO 1976-03-06T23:59:59.999Z+1YEAR]` • `lastAccessed:[1976-03-06T23:59:59.999Z/YEAR TO 1976-03-06T23:59:59.999Z]`

Dataset Properties

Name	Type	Description
`compressionType`	tokenizedCaseInsensitiveText	The type of compression of a dataset file.
`dataType`	string	The data type: record.
`datasetType`	tokenizedCaseInsensitiveText	The type of the dataset: Kite.
`fileFormat`	tokenizedCaseInsensitiveText	The format of a dataset file: Avro or Parquet.
`fullDataType`	string	The full data type: record.

Name	Type	Description
partitionType	string	The type of the partition.
schemaName	string	The name of the dataset schema.
schemaNameSpace	string	The namespace of the dataset schema.

HDFS Properties

Name	Type	Description
blockSize	long	The block size of an HDFS file.
deleted	Boolean	Indicates whether the entity has been moved to the Trash folder.
deleteTime	date	The time the entity was moved to the Trash folder.
fileSystemPath	path	The path to the entity.
mimeType	ngramedText	The MIME type of an HDFS file.
parentPath	string	The path to the parent entity of a child entity. For example: `parent path:/default/sample_07` for the table `sample_07` from the Hive database `default`.
permissions	string	The UNIX access permissions of the entity.
replication	int	The number of copies of HDFS file blocks.
size	long	The exact size of the entity in bytes or a range of sizes. Range examples: `size:[1000 TO *]`, `size: [* TO 2000]`, and `size:[* TO *]` to find all fields with a size value.

Hive Properties

Name	Type	Description
Field		
dataType	ngramedText	The type of data stored in a field (column).
Table		
compressed	Boolean	Indicates whether a table is compressed.
serDeLibName	string	The name of the library containing the SerDe class.
serDeName	string	The fully qualified name of the SerDe class.
Partition		
partitionColNames	string	The table columns that define the partition.
partitionColValues	string	The table column values that define the partition.
technical_properties	string	Hive extended attributes.
clusteredByColNames	string	The column names that identify how table content is divided into buckets.
sortByColNames	string	The column names that identify how table content is sorted within a bucket.

Cloudera Navigator Metadata Architecture

MapReduce and YARN Properties

Name	Type	Description
inputRecursive	Boolean	Indicates whether files are searched recursively under the input directories, or only files directly under the input directories are considered.
jobId	ngramedText	The ID of the job. For a job spawned by Oozie, the workflow ID.
mapper	string	The fully qualified name of the mapper class.
outputKey	string	The fully qualified name of the class of the output key.
outputValue	string	The fully qualified name of the class of the output value.
reducer	string	The fully qualified name of the reducer class.

Operation Properties

Name	Type	Description
Operation		
inputFormat	string	The fully qualified name of the class of the input format.
outputFormat	string	The fully qualified name of the class of the output format.
Operation Execution		
inputs	string	The name of the entity input to an operation execution. For entities of resource type mapreduce, yarn, and spark, it is usually a directory. For entities of resource type hive, it is usually a table.
outputs	string	The name of the entity output from an operation execution. For entities of resource type mapreduce, yarn, and spark, it is usually a directory. For entities of resource type hive, it is usually a table.
engineType	string	The type of the engine used for an operation: MR or Spark.

Oozie Properties

Name	Type	Description
status	string	The status of the Oozie workflow: RUNNING, SUCCEEDED, or FAILED.

Pig Properties

Name	Type	Description
scriptId	string	The ID of the Pig script.

Sqoop Properties

Name	Type	Description
dbURL	string	The URL of the database from or to which the data was imported or exported.
dbTable	string	The table from or to which the data was imported or exported.
dbUser	string	The database user.
dbWhere	string	A where clause that identifies which rows were imported.

Name	Type	Description
dbColumnExpression	string	An expression that identifies which columns were imported.

Accessing Metadata

Minimum Required Role: Metadata Viewer (also provided by **Metadata Administrator, Full Administrator**)

You can access metadata through the Navigator UI or through the Navigator API.

Navigator Metadata UI

Searching Metadata

1. Start and log into the Cloudera Navigator data management component UI.
2. Do one of the following:

   - Type a search string into the **Search** box that conforms to the search syntax.
   - Click the **Click here** link.

   The Search page displays. It has a Search box and two panes: the Filters pane and the Search Results pane.

To display all entities, click **Clear all filters**. You filter the search results by specifying filters or typing search strings in the Search box.

Filter Example

The following filter example demonstrates how to narrow search results by selecting a built-in Source Type filter set to HDFS and the managed property emailFrom with the value terry@example.com.
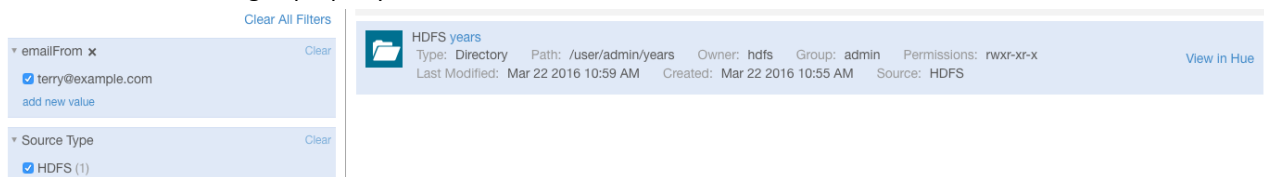
Search Results

The Search Results pane displays the number of matching entries in pages listing 25 entities per page. You can view the pages using the page control
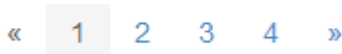
at the bottom of each page.

Each entry in the result list contains:

- Source type
- Name - A link to a page that displays the entity details and lineage diagram
- Properties
- If Hue is running, a link at the far right labeled **View in Hue** that opens the Hue browser for the entity:

  - HDFS directories and files - File Browser
  - Hive database and tables - Metastore Manager
  - MapReduce, YARN, Pig - Job Browser

For example:

Cloudera Navigator Metadata Architecture

Hive **sample_07**
Type: **Table**   Parent Path: /default   Path: hdfs://tcdn-1.gce.cloudera.com:8020/user/hive/warehouse/sample_07   Owner: hdfs   View in Hue
Last Modified: Mar 29 2016 2:53 PM   Created: Mar 29 2016 11:22 AM   Source: Hive

Displaying Entity Details

The entity Details page displays all three types of metadata for an entity—technical, managed, and custom—and entity type-specific information:

- HDFS directories - Directory contents
- HDFS datasets and fields - Schema
- Hive and Impala databases - Tables and views
- Hive tables - Extended attributes, table schema, partitions
- Impala tables - Table schema
- MapReduce, YARN, and Spark operations - Instances
- Pig operation executions - Tables

All entity types can display inputs and outputs. See Configuring Display of Inputs and Outputs.

If managed properties have been defined for a particular entity type, the **Show All** checkbox in the Managed Metadata pane displays all properties that can be assigned values for the selected entity. To display only those properties that have values, clear the checkbox. If all properties have values, the checkbox has no effect.

To display entity details:

1. Perform a search.
2. In the search results, click an entity name link. The Details tab displays.

Hive Table Custom Metadata Example

For example, if you click the Hive table `sample_07` link in the search result displayed in the preceding section, you see the following details:

sample_07   Actions ▾   Details   Lineage

Technical Metadata

Source Type:	HIVE
Type:	Table
Parent Path:	/default
Path:	hdfs://tcdn-1.gce.cloudera.com:...
Compressed:	false
SerDe Library:	org.apache.hadoop.hive.serde2.l...
Input Format:	org.apache.hadoop.mapred.Text...
Output Format:	org.apache.hadoop.hive.ql.io.Hiv...
Owner:	hdfs
Last Modified:	Mar 29 2016 2:53 PM
Created:	Mar 29 2016 11:22 AM
Source:	Hive
Class Name:	hv_table
Package Name:	nav

Managed Metadata

No metadata available

Custom Metadata

Tags:	tag1, tag2
customkey:	value

Hive Extended Attributes

key1:	value1

Schema

- code string
- description string
- salary int
- total_emp int

In addition to the technical metadata, this Hive table has custom metadata consisting of tags `tag1` and `tag2`, a custom key-value pair `customkey-value`, and an extended Hive attribute key-value pair `key1-value1`. The Details page also displays the table schema.

File-Managed Metadata Example

The following `years` directory entity has two managed properties in the `MailAnnotation` namespace: `emailFrom` and `emailTo`. The former is single-valued and the latter is multivalued.
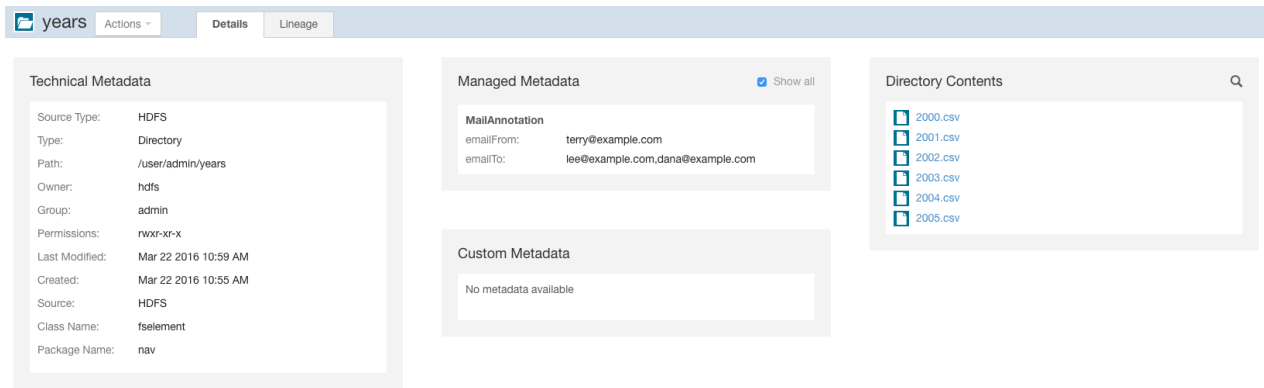
In the following example, the **Show All** checkbox is selected. The `emailFrom` property is configured, but the `emailTo` property is not configured:

When the **Show All** checkbox is cleared, only the configured `emailFrom` property displays:

Filtering Search Results

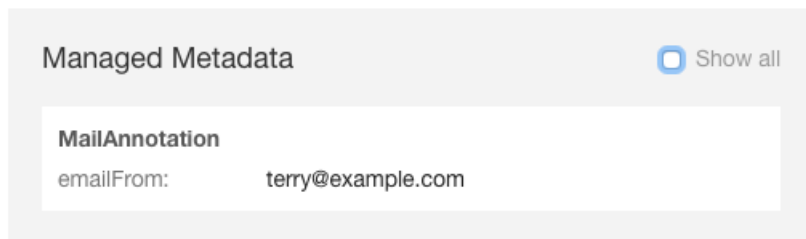To filter search results, specify filters in the Filters pane or type search strings in the **Search** box.

The Filters pane contains a set of default properties (source type, type, owner, cluster, and tags) and property values (also called *facets*). You can add a filter by clicking in **Add Another Filter...** and scrolling or by typing in the filter combo box to search for it.

As you add filters, filter breadcrumbs are added between Search box and search results, and search results are refreshed immediately. Multiple filters composed with the + operator are separated with the | character.

To remove nondefault filter properties, click the in the filter.

Specify a property value as follows:

- **Boolean** - Select the option to respectively not display, or display only those entries, with the value set to true: **Do not show *XXX*** (the default) or **Show *XXX* only**, where *XXX* is the Boolean property.
- **Enumerated or freeform string**

  – Select the checkbox next to a value or click a value link.
  – If a property has no values, click **add a new value**, click the text box, and select from the populated values in the drop-down list or type a value.

- **Timestamp** - Timestamps are used for started, ended, created, last accessed, and last modified properties. The server stores the timestamp in UTC, and the UI displays the timestamp converted to the local timezone. Select one of the timestamp options:

  - A **Last *XXX* day(s)** link.
  - The **Last** checkbox, type or specify the value using the spinner control [icon] and select the unit minutes, hours, or days.
  - The **Custom period** checkbox and specify the start and end date.

    - Date - Click the down arrow [icon] to display a calendar and select a date, or click a field and click the spinner arrows [icon] or up and down arrow keys.
    - Time - Click the hour, minute, and AM/PM fields and click the spinner arrows [icon] or up and down arrow keys to specify the value.
  - Move between fields by clicking fields or by using the right and left arrow keys.

To remove filter values, click the [icon] in the breadcrumb or clear the checkbox.

When you select a specific source type value, additional properties that apply to that source type display. For example, HDFS has size, created, and group properties:

The number in parentheses (facet count) after a property value is the number of extracted entities that have that property value:

▾ Source Type
☑ HDFS (2529)
　　▸ Size (MiB)
　　▸ Created
　　▸ Group
☑ Hive (22)
　　▸ Started
　　▸ Ended
☐ Impala (17)

Facet values with the count 0 are not displayed.

When you type values, the value is enclosed in quotes; the value inside the quotes must match the metadata exactly. For example:

- Typing "sample_*" in the `originalName` property returns only entities whose names match that exact string.
- To perform a wildcard search, type the wildcard string in the Search box. For example, typing the string "sample_*" in the Search box returns all entities with "sample_" at the beginning of their original name.

When you construct search strings with filters, use parentheses `()` to specify multiple values of a property. Add multiple properties by using the + operator. For example, entities of type HDFS or Hive that are of type file or directory:

```
+(sourceType:hdfs sourceType:hive) +(type:file type:directory)
```

and:

```
((+sourceType:hdfs +created:[NOW/DAY-30DAYS TO NOW/DAY+1DAY]) sourceType:hive)
```

Saving Searches

1. Specify a search string or set of filters.
2. To the right of the Search box, select **Actions** > **Save**, **Actions** > **Save** *Search_name*, or **Actions** > **Save As...**.
3. If you have not previously saved the search, specify a name and click **Save**.

Reusing a Saved Search

1. To the right of the Search box, select **Actions** > **View saved searches...**. A label with the saved search name is added under the Search box.
2. Click the saved search name. The breadcrumbs and full query if displayed are updated to reflect the saved search, and the search results are refreshed immediately.

Navigator Metadata API

The Navigator API allows you to search entity metadata using a REST API. For information about the API, see Cloudera Navigator Data Management API.

Defining Managed Metadata

Minimum Required Role: Metadata Administrator (also provided by **Full Administrator**)

You can use managed metadata to add typed metadata to classes of entities. You can add namespaces and properties.

A **namespace** is a container for properties. Four namespaces are reserved:

- `nav` for Navigator metadata classes (for example, `fselement` and user-defined custom fields)

- `up` ([custom metadata](#))
- `tp` (technical properties)
- `xt` (partner applications)

The combination of namespace and property name must be unique.

A property can be one of the following types:

- Boolean
- date
- integer
- long
- float
- double
- text (with optional maximum length and regular expression validation criteria)
- enum (of string)

A property can be single-valued or assume multiple values.

Once you have created properties and assigned property values to specific entities, you can create [search filters](#) for property values.

Creating Custom Properties Using the Metadata UI

> **Important:** You cannot delete or modify custom namespaces, properties, and enumeration values. Once you have created a custom property, you can only add classes to which the property applies, and enumeration values. The creation and edit property wizards require you to review the property before creating or editing it.

To create custom properties:

1. [Start and log into the Cloudera Navigator data management component UI](#) with the credentials of a user having one of the following user roles:

   - Cloudera Manager Full Administrator
   - Cloudera Manager Navigator Administrator
   - Cloudera Navigator Full Administrator
   - Cloudera Navigator Metadata Administrator

2. Click the **Administration** link in the upper right. The **Managed Metadata** tab displays the list of namespaces and the properties defined in the namespaces.
3. Click the **Create...** button.
4. In the Class field, click

   ▾

   or type the beginning of a [Navigator entity classname](#).

5. Select the class of entities to which the property applies. To clear the field, hover over the field and click the ⊗ that displays at the right of the field.
6. Click the Namespace field and select a namespace. If the Namespace drop-down list is empty, click **Create Namespace...**.

   a. Specify a namespace name and optional description.
   b. Click **Continue**.

7. Add the name for the property.
8. Specify an optional description.
9. Select the **Multivalued Enable** checkbox if the property can have more than one value. For example, an `emailFrom` property should accept only one value, but an `emailTo` property could accept more than one value.
10. In the **Type** drop-down list, select the property type and specify constraints on the value.

- **Boolean** - Boolean: true or false.
- **Date** - Date and time.
- **Enum** - A set of values. In the **Enumeration** field, type valid enumeration values and press **Enter** or **Tab**.
- **Number** - A number. In the **Number Type** field, select the type of the number: **Integer**, **Long**, **Float**, **Double**.
- **Text** - A string.

    - **Maximum Length** - The maximum length of the string.
    - **Regular Expression** - A regular expression that determines whether a string is a valid value.

11. Click **Continue to Review**. The Review and Save screen displays.
12. Click **Create** to commit the change or **Back to Edit Property** to continue editing the property.

Example Properties

The following figure shows two properties in the namespace `MailAnnotation` that apply to entities of the `fselement` class (that is, HDFS files and directories). The `emailFrom` property is of type TEXT and can be assigned a single value. The `emailTo` property is also of type TEXT but can have multiple values.

Properties

See the documentation before creating or modifying custom properties.

Name	Type	Type Specification	Class	Description	Multivalued
MailAnnotation					
emailFrom	TEXT	Regex: \b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Z...	nav.fselement	Annotates a file with the sender of an email message.	No
emailTo	TEXT	Regex: \b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Z...	nav.fselement	Annotates a file with the recipients of an email messag...	Yes

Editing Custom Properties Using the Metadata UI

The *only changes* you can make to a custom property are to add classes to which the property applies and enumeration values.

1. Start and log into the Cloudera Navigator data management component UI with the credentials of a user having one of the following roles:

    - Cloudera Manager Full Administrator
    - Cloudera Manager Navigator Administrator
    - Cloudera Navigator Full Administrator
    - Cloudera Navigator Metadata Administrator

2. In the Properties table, click a property link. The Edit *propertyname* dialog box displays.
3. In the Additional Class field, click

    ▾

    or type the beginning of a Navigator entity classname.
4. Select the class of entities to which the property applies. To clear the field, hover over the field and click the ⊗ that displays at the right of the field.
5. In the Additional Enumeration Values field, type valid enumeration values and press **Enter** or **Tab**.
6. Click **Continue to Review**. The Review and Save screen displays.
7. Click **Update** to commit the change or **Back to Edit Property** to continue editing the property.

Navigator Built-in Classes

Class	Description
hdfs_dataset	Logical dataset backed by a path in HDFS.

Class	Description
hdfs_dataset_field	Field in an HDFS dataset.
fselement	DFS file or directory.
hv_column	Column in a Hive table.
hv_database	Hive database.
hv_partition	Partition of a Hive table.
hv_query	Hive query template.
hv_query_exec	Instance of a Hive query.
hv_query_part	Component of a Hive query that maps specific input columns to output columns.
hv_table	A Hive table.
hv_view	View on one or more Hive tables.
impala_query	Impala query template.
impala_query_exec	Instance of an Impala query.
impala_query_part	Component of an Impala query that maps specific input columns to output columns.
mrjobinstance	Instance of a MapReduce, YARN, or Spark job.
mrjobspec	Template for a MapReduce, YARN, or Spark job.
oozie_workflow	Template for an Oozie workflow.
oozie_wf_instance	Instance of an Oozie workflow.
sq_export_sub_oper	Sqoop export component that connects specific columns.
sq_qry_import	Sqoop import job with query options.
sq_import_sub_oper	Sqoop import component that connects specific columns.
sq_oper_exec	Instance of a Sqoop job.
sq_tbl_import	Sqoop table import operation template.
sq_tbl_export	Sqoop table export operation template.
pig_field	Field for a relation in Pig; similar to a column in a Hive table.
pig_operation	Template for a Pig transformation.
pig_op_exec	Instance of a Pig transformation.
pig_relation	Pig relation; similar to a Hive table.
userexpression	User-specified sub-operation of a MapReduce or YARN job; used for specifying custom column-level lineage.

Defining Metadata with the Navigator API and Navigator SDK

In addition to defining metadata using features provided by the Navigator Metadata UI, you can also define metadata using the Navigator API and Navigator SDK.

For information on the Navigator API, see Cloudera Navigator Data Management API.

For information on the SDK, see the Navigator SDK documentation.

Adding and Editing Metadata

Minimum Required Role: Metadata Administrator (also provided by **Full Administrator**)

Cloudera Navigator supports adding metadata to extracted entities. You can add and edit two types of metadata:

- Custom metadata - Display name, description, tags, and key-value pairs. You can add and edit custom metadata using the Navigator UI, MapReduce service and job properties, HDFS metadata files, and the Cloudera Navigator Data Management API.
- Managed metadata. You can add and edit managed metadata using the Navigator UI and the API.

Adding and Editing Metadata Using the Navigator UI

1. Run a search in the Navigator UI.
2. Click an entity link returned in the search. The Details tab displays.
3. To the left of the Details tab, click **Actions** > **Edit Metadata...**. The Edit Metadata dialog box drops down.
4. Add metadata fields:

    - In the Name field, type a new display name.
    - In the Description field, type a description.
    - In the Tags field, type a tag and press **Enter** or **Tab** to create new tag entries.
    - **Managed Metadata**

        1. Click the

            ▾

            and select a property.
        2. Click the value field after the ⁚ to display type-specific selection controls such as integer spinners and date selection controls. Either type the value or use the controls to select a value.

            +∢

            to add another managed property key-value pair or another value for a given key.
    - **Key-Value Pairs**

        1. Click

            +

            to add a key-value pair.
        2. Type a key and a value. You can specify special characters (for example, ".", " ") in the name, but it makes searching for the entity more difficult because some characters collide with special characters in the search syntax.

    > **Note:** You cannot assign managed metadata in the Key-Value Pairs field because you cannot specify the namespace. All properties specified in the Key-Value Pairs field are treated as custom metadata.

5. Click **Save**. The new metadata appears in the Managed Metadata or Custom Metadata pane.

Custom Metadata Example

In the following figure, the tag `northeast` and the property year with value 2016 have been added to the file `customers`:

After you save, the metadata appears in the Custom Metadata pane:

Managed Metadata Example

In the following figure, managed properties are being configured for the `years` entity. The managed property `emailFrom` has been configured with a value, and the managed property `emailTo` is having a second value configured:

Name

years

Description

Managed Metadata

emailFrom   ▼ : terry@example.com   −

emailTo   ▼ : lee@example.com   −

              ▲ :   − +

MailAnnotation
  emailTo

If a provided value does not satisfy the constraints specified in the property definition—in this case, a regular expression that describes email addresses—a message displays when you try to save the value:

emailTo   ▼ : jane   − +

⚠ The value needs to match format
\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Za-z0-9])@(([A-Za-z0-9]|[A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])\.)+([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b
The value needs to match format \b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._]*[A-Za-z0-9])@(([A-Za-z][A-Za-z0-9\-]*[A-Za-z0-9])\.)+([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b

Tags

Key-Value Pairs

+

After you correctly specify the second value and save, the properties display in the Managed Metadata pane:
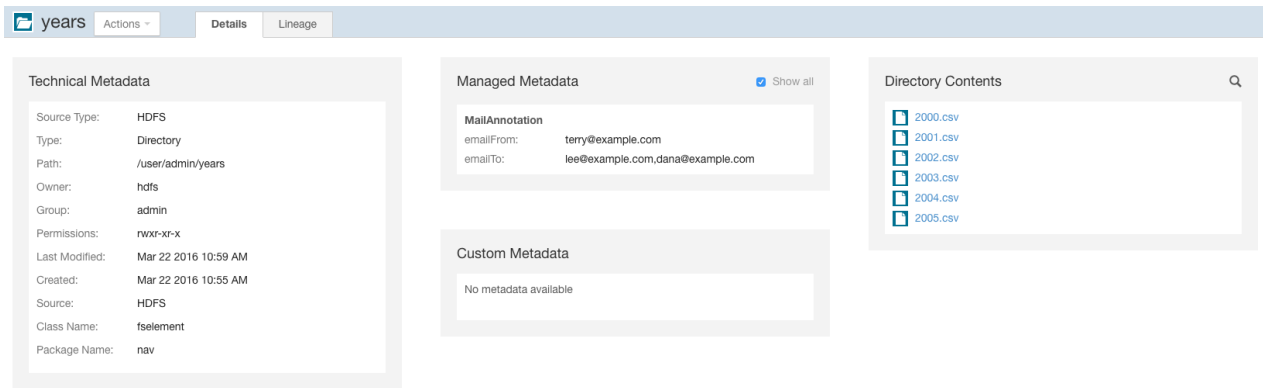
Editing MapReduce Custom Metadata

You can associate custom metadata with arbitrary configuration parameters to MapReduce jobs and job executions. The configuration parameters to be extracted by Navigator can be specified statically or dynamically.

To specify configuration parameters statically for all MapReduce jobs and job executions, do the following:

1. Do one of the following:

   - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
   - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the **Configuration** tab.
3. Select **Scope** > **Navigator Metadata Server**.
4. Select **Category** > **Advanced**.
5. Click **Navigator Metadata Server Advanced Configuration Snippet for cloudera-navigator.properties**.
6. Specify values for the following properties:

   - `nav.user_defined_properties` - A comma-separated list of user-defined property names.
   - `nav.tags` - A comma-separated list of property names that serve as tags. The property `nav.tags` can point to multiple property names that serve as tags, but each of those property names can only specify a *single* tag.

7. Click **Save Changes** to commit the changes.
8. Click the **Instances** tab.
9. Restart the role.
10. In the MapReduce job configuration, set the value of the property names you specified in step 6.

To specify configuration parameters dynamically:

1. Specify one or more of the following properties in a job configuration:

   - Job properties (`type:OPERATION`)

     - `nav.job.user_defined_properties` - A comma-separated list of user-defined property names
     - `nav.job.tags` - A comma-separated list of property names that serve as tags

   - Job execution properties (`type:OPERATION_EXECUTION`)

     - `nav.jobexec.user_defined_properties` - A comma-separated list of user-defined property names
     - `nav.jobexec.tags` - A comma-separated list of property names that serve as tags

   The properties `nav.job.tags` and `nav.jobexec.tags` can point to multiple property names that serve as tags, but each of those property names can only specify a *single* tag.

2. In the MapReduce job configuration, set the value of the property names you specified in step 1.

Example: Setting Properties Dynamically

Add the tags `onetag` and `twotag` to a job:

1. Dynamically add the `job_tag1` and `job_tag2` properties:

```
conf.set("nav.job.tags", "job_tag1, job_tag2");
```

2. Set the `job_tag1` property to `onetag`:

```
conf.set("job_tag1", "onetag");
```

3. Set the `job_tag2` property to `twotag`:

```
conf.set("job_tag2", "twotag");
```

Add the tag `atag` to a job execution:

1. Dynamically add the `job_tag` property:

```
conf.set("nav.jobexec.tags","job_exec_tag");
```

2. Set the `job_exec_tag` property to `atag`:

```
conf.set("job_exec_tag", "atag");
```

Add the user-defined key `key` with the value `value`:

1. Dynamically add the user-defined key `bar`:

```
conf.set("nav.job.user_defined_properties", "key");
```

2. Set the value of the user-defined key `key` to `value`:

```
conf.set("key", "value")
```

Editing HDFS Custom Metadata Using Metadata Files

You can add tags and properties to HDFS entities using metadata files. With metadata files, you can assign metadata to entities in bulk and create metadata before it is extracted. A metadata file is a JSON file with the following structure:

```
{
  "name" : "aName",
  "description" : "a description",
  "properties" : {
    "prop1" : "value1", "prop2" : "value2"
  },
  "tags" : [ "tag1" ]
}
```

To add metadata files to files and directories, create a metadata file with the extension `.navigator`, naming the files as follows:

- **File** - The path of the metadata file must be `.filename.navigator`. For example, to apply properties to the file `/user/test/file1.txt`, the metadata file path is `/user/test/.file1.txt.navigator`.
- **Directory** - The path of the metadata file must be *dirpath*`/.navigator`. For example, to apply properties to the directory `/user`, the metadata path must be `/user/.navigator`.

The metadata file is applied to the entity metadata when the extractor runs.

Editing HDFS and Hive Metadata Using the Navigator Metadata API

You can use the Cloudera Navigator Data Management API to modify the custom metadata of HDFS or Hive entities, whether the entities have been extracted or not. If an entity has been extracted when the API is called, the metadata is applied immediately. If the entity has not been extracted, you can preregister metadata, which is then applied once the entity is extracted. Metadata is saved regardless of whether or not a matching entity is extracted, and Navigator does not perform any cleanup of unused metadata.

If you call the API before the entity is extracted, the custom metadata is stored with the entity's:

- Identity
- Source ID
- Metadata fields (name, description, tags, properties)
- Fields relevant to the identifier

The rest of the entity fields (such as type) are not present. To view all stored metadata, use the API to search for entities without an internal type:

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/?query=-internalType:*
 -u username:password -X GET
```

Custom metadata provided through the API overwrites existing metadata. For example, if you call the API with an empty name and description, empty array for tags, and empty dictionary for properties, the call removes this metadata. If you omit the tags or properties fields, the existing values remain unchanged.

Modifying custom metadata using HDFS metadata files and the metadata API at the same time *is not* supported. You must use one or the other, because the two methods work differently. Metadata specified in files is merged with existing metadata, whereas the API overwrites metadata. Also, the updates provided by metadata files wait in a queue before being merged, but API changes are committed immediately. Some inconsistency can occur if a metadata file is merged when the API is in use.

You modify metadata using either the PUT or POST method. Use the PUT method if the entity has been extracted, and the POST method to preregister metadata. Use the following syntax:

- PUT

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/identity -u
username:password -X PUT -H\
"Content-Type: application/json" -d '{properties}'
```

where *identity* is an entity ID and *properties* are:

- name - Name metadata.
- description - Description metadata.
- tags - Tag metadata.
- properties - Custom metadata properties. The format is {key: value}.
- customProperties - Managed metadata properties. The format is {namespace: {key: value}}. If a property is assigned a value that does not conform to type constraints, an error is returned.

All existing naming rules apply, and if any value is invalid, the entire request is denied.

- POST

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/ -u username:password
-X POST -H\
"Content-Type: application/json" -d '{properties}'
```

where *properties* are:

– <u>sourceId</u> (required) - An existing source ID. After the first extraction, you can retrieve source IDs using the call:

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/?query=type:SOURCE -u
username:password -X GET
```

For example:

```
[ ...
{  {
"identity": "61cfefd303d4284b7f5014b701f2c76d",
"originalName": "source.listing",
"originalDescription": null,
"sourceId": "012437f9eeb3c23dc69e679ac94a7fa2",
"firstClassParentId": null,
"parentPath": "/user/hdfs/.cm/distcp/2016-02-03_487",
...
"properties": {
"__cloudera_internal__hueLink":
"http://vc0142.halxg.cloudera.com:8888/filebrowser/#/user/hdfs/.cm/distcp/2016-02-03_487/source.listing"

 },
"technicalProperties": null,
"fileSystemPath": "/user/hdfs/.cm/distcp/2016-02-03_487/source.listing",
"type": "FILE",
"size": 92682,
"created": "2016-02-03T21:12:16.587Z",
"lastModified": "2016-02-03T21:12:16.587Z",
"lastAccessed": "2016-02-03T21:12:16.587Z",
"permissions": "rw-r--r--",
"owner": "hdfs",
"group": "supergroup",
"blockSize": 134217728,
"mimeType": "application/octet-stream",
"replication": 3,
"userEntity": false,
"deleted": false,
"sourceType": "HDFS",
"metaClassName": "fselement",
"packageName": "nav",
"internalType": "fselement"
}, ...
```

If you have multiple services of a given type, you must specify the source ID that contains the entity you expect it to match.

– `parentPath` - The path of the parent entity, defined as:

  – HDFS file or directory - `fileSystemPath` of the parent directory. (Do not provide this field if the entity affected is the root directory.) Example `parentPath` for `/user/admin/input_dir`: `/user/admin`. If you add metadata to a directory, the metadata does not propagate to any files or folders in that directory.
  – Hive database - If you are updating database metadata, do not specify this field.
  – Hive table or view - The name of database containing the table or view. Example for a table in the default database: `default`.
  – Hive column - *database name/table name/view name*. Example for a column in the sample_07 table: `default/sample_07`.

– `originalName` (required) - The name as defined by the source system.

  – HDFS file or directory- Name of file or directory (`ROOT` if the entity is the root directory). Example `originalName` for `/user/admin/input_dir`: `input_dir`.
  – Hive database, table, view, or column - The name of the database, table, view, or column.

    – Example for default database: `default`
    – Example for sample_07 table: `sample_07`

- – `name` - Name metadata.
- – `description` - Description metadata.
- – `tags` - Tag metadata.
- – `properties` - Custom metadata properties. The format is {key: value}.

All existing naming rules apply, and if any value is invalid, the entire request is denied.

HDFS PUT Custom Metadata Example for /user/admin/input_dir Directory

```
curl
http://Navigator_Metadata_Server_host:port/api/v9/entities/e461de8de38511a3ac6740dd7d51b8d0
 -u username:password -X PUT -H "Content-Type: application/json"\
-d '{"name":"my_name","description":"My description",
"tags":["tag1","tag2"],"properties":{"property1":"value1","property2":"value2"}}'
```

HDFS POST Custom Metadata Example for /user/admin/input_dir Directory

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/ -u username:password
-X POST -H "Content-Type: application/json"\
-d '{"sourceId":"a09b0233cc58ff7d601eaa68673a20c6",
"parentPath":"/user/admin","originalName":"input_dir", "name":"my_name","description":"My
 description",\
"tags":["tag1","tag2"],"properties":{"property1":"value1","property2":"value2"}}'
```

Hive POST Custom Metadata Example for total_emp Column

```
curl http://Navigator_Metadata_Server_host:port/api/v9/entities/ -u username:password
-X POST -H "Content-Type: application/json"\
-d '{"sourceId":"4fbdadc6899638782fc8cb626176dc7b",
"parentPath":"default/sample_07","originalName":"total_emp",\
"name":"my_name","description":"My description",
"tags":["tag1","tag2"],"properties":{"property1":"value1","property2":"value2"}}'
```

HDFS PUT Managed Metadata Example

The following example demonstrates how to set two properties in the `MailAnnotation` namespace: a multivalued property `emailTo` and a single-valued property `emailFrom`:

```
curl
http://Navigator_Metadata_Server_host:port/api/v9/entities/87afcb92d5de856c7e8292e2e12cf1ea
 -u admin:admin -X PUT -H "Content-Type: application/json"
-d
'{"customProperties":{"MailAnnotation":{"emailTo":["lee@example.com","dana@example.com"],"emailFrom":"terry@email.com"}}}'
```

The response is:

```
{
  "identity" : "87afcb92d5de856c7e8292e2e12cf1ea",
  "originalName" : "years",
  "originalDescription" : null,
  "sourceId" : "012437f9eeb3c23dc69e679ac94a7fa2",
  "firstClassParentId" : null,
  "parentPath" : "/user/admin",
  "extractorRunId" : "012437f9eeb3c23dc69e679ac94a7fa2##1",
  "customProperties" : {
    "MailAnnotation" : {
      "emailTo" : [ "lee@example.com", "dana@example.com" ],
      "emailFrom" : "terry@email.com"
    }
  },
  "name" : null,
  "description" : null,
  "tags" : null,
  "properties" : {
   "__cloudera_internal__hueLink" : "Hue_Server_host:8888/filebrowser/#/user/admin/years"

  },
```

```
   "technicalProperties" : null,
   "fileSystemPath" : "/user/admin/years",
   "type" : "DIRECTORY",
   "size" : null,
   "created" : "2016-03-22T17:55:31.902Z",
   "lastModified" : "2016-03-22T17:59:14.065Z",
   "lastAccessed" : null,
   "permissions" : "rwxr-xr-x",
   "owner" : "hdfs",
   "group" : "admin",
   "blockSize" : null,
   "mimeType" : null,
   "replication" : null,
   "sourceType" : "HDFS",
   "metaClassName" : "fselement",
   "userEntity" : false,
   "deleted" : false,
   "packageName" : "nav",
   "internalType" : "fselement"
}
```

Accessing and Editing Metadata with the Navigator SDK

In addition to the metadata features provided by the Cloudera Navigator data management component UI, you can also access and edit metadata using the Navigator SDK. For information on the SDK, see the Navigator SDK documentation.

Performing Actions on Entities

Minimum Required Role: Policy Administrator (also provided by **Full Administrator**)

Moving an HDFS Entity and Moving an HDFS Entity to Trash

You can move an HDFS entity to another location, and to HDFS trash. To perform such actions, you must be a member of a user group that has the appropriate access to HDFS files.

You can also schedule a move or move to trash in a policy.

1. Start and log into the Cloudera Navigator data management component UI.
2. Run a search in the Navigator UI.
3. Click an HDFS entity link returned in the search. The entity Details tab displays.
4. To the left of the Details tab, select **Actions** > **Move...** or **Actions** > **Move to Trash...**.
5. For a move, specify the target path.
6. Click **Run Action**. When you delete a file, after a short delay the file displays a Deleted badge.

Viewing Command Action Status

1. Start and log into the Cloudera Navigator data management component UI.
2. In the top right, click *username* > **Command Actions**. The Command Actions status page displays with a list of actions performed and the policy that caused the action, if applicable.
3. If an action failed, a View Log button displays, which you can click to view the error message associated with the failure.

Viewing an Entity in Hue

If you are running Hue, you can view certain types of entities in the following Hue applications:

- HDFS directories and files - File Browser
- Hive database and tables - Metastore Manager
- MapReduce, YARN, Pig - Job Browser

1. Start and log into the Cloudera Navigator data management component UI.

2. Run a search in the Navigator UI.
3. Do one of the following:

   - Search results

      1. Click the **View in Hue** link in a search result entry.

   - Entity details

      1. Click an entity link returned in the search. The entity Details tab displays.
      2. To the left of the Details tab, select **Actions** > **View in Hue**.

   The entity displays in the supported Hue application.

Cloudera Navigator Provenance Use Case

A number of business decisions and transactions rely on the verifiability of the data used in those decisions and transactions. Data-verification questions might include:

- How was this mortgage credit score computed?
- How can I prove that this number on a sales report is correct?
- What data sources were used in this calculation?

You can use Cloudera Navigator to answer these and other questions about your data. Using metadata and lineage, you can get track the life of the data to verify its ***provenance***—that is, determine its origin.

How Can I Verify a Value in a Table?

A number of business transactions require you to verify that information is correct and that it is derived from a reliable source. For example, if you work in a sales orgainization, you might verify that information in sales reports is accurate, that you can trust the contents, and that you can identify the origin of the information.

The following example shows how you can verify information in a field named **s_neighbor** by tracing it to its source. You will replace the fields and other information in this example with the actual information that you want to verify.

1. Log into the Cloudera Navigator data management UI and click the **Search** tab.
2. Type **s_neighbor** in the search box.

You see four instances of the **s_neighbor** field.

3. View details of the field in the **top_10 table** by clicking **s_neighbor** in the entry with the Parent Path /default/top10.

You see that the parent table is **top_10**, and the input or upstream source of the data is the **salesdata** database.

Where did **salesdata** come from originally? It was imported using sqoop, with syntax similar to the following; actual arguments vary:

```
> sqoop import-all-tables
    -m {{cluster_data.worker_node_hostname.length}} \
    --connect jdbc:mysql://{{cluster_data.manager_node_hostname}}:3306/retail_db \
    --username=admin \
    --password=password \
    --compression-codec=snappy \
    --as-parquetfile \
    --warehouse-dir=/user/hive/warehouse \
    --hive-import
```

4. To see a graphical representation of the relationships among the entities:

   a. Click the **Lineage** tab.
   b. In Lineage Options, select **Operations** and clear any other check boxes.

See that **s_neighbor** can be traced back to the original table **salesdata**.

5. Click the operation entity in the center of the lineage diagram, and see details about it on the lower right side of the lineage window.

Information about the selected entity indicates that the operation is an Impala query. Click the information icon on the Query Text line to see the entire query. This query was used to derive **top_10** from the original table.

Cloudera Navigator Auditing Architecture

Cloudera Navigator auditing provides data auditing and access features. The Cloudera Navigator auditing architecture is illustrated below.
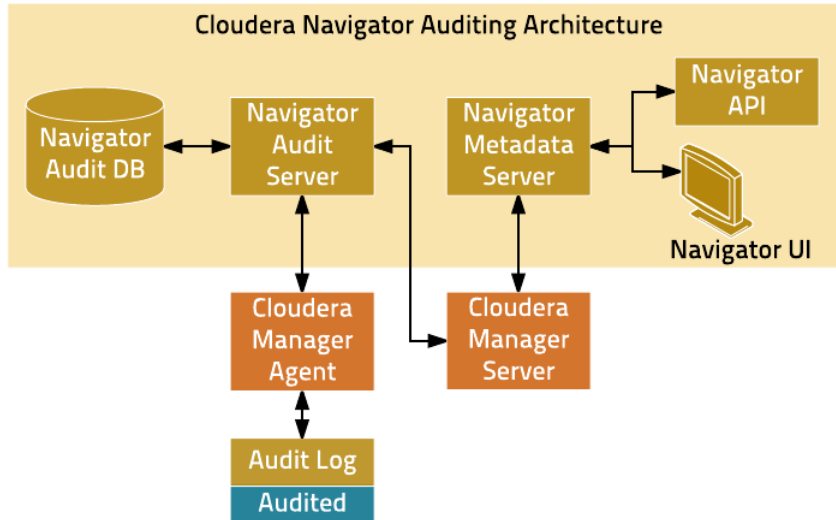
When Cloudera Navigator auditing is configured, plug-ins that enable collection and filtering of service access events are added to the HDFS, HBase, and Hive (that is, the HiveServer2 and Beeswax servers) services. The plug-ins write the events to an audit log on the local filesystem. The existence of the plug-ins places requirements on these services when Cloudera Navigator is upgraded. Cloudera Impala, Sentry, and the Cloudera Navigator Metadata Server collect and filter access events and write them to an audit log file.

The Cloudera Manager Agent monitors the audit log files and sends the events to the Navigator Audit Server. The Cloudera Manager Agent retries any event that it fails to transmit. As there is no in-memory transient buffer involved, once the events are written to the audit log file, they are guaranteed to be delivered (as long as filesystem is available). The Cloudera Manager Agent keeps track of current event offset in the audit log that it has successfully transmitted, so on any crash/restart it picks up the event from the last successfully sent position and resumes. Audit logs are rotated and the Cloudera Manager Agent follows the rotation of the log. The Agent also takes care of purging old audit logs once they have been successfully transmitted to the Navigator Audit Server. If a plug-in fails to write an event to the audit log file, it can either drop the event or shut down the process in which they are running (depending on the configured queue policy).

The Navigator Audit Server performs the following functions:

- Tracking and coalescing events
- Storing events to the audit database

Cloudera Navigator Auditing

Minimum Required Role: Auditing Viewer (also provided by **Full Administrator**)

An **audit event** is an event that describes an action that has been taken for a cluster, host, license, parcel, role, service or user.

Cloudera Manager records cluster, host, license, parcel, role, and service **lifecycle events** (activate, create, delete, deploy, download, install, start, stop, update, upgrade, and so on), user **security-related events** (add and delete user, login failed and succeeded), and provides an audit UI and API to view, filter, and export such events. For information on Cloudera Manager auditing features, see Lifecycle and Security Auditing.

The Cloudera Navigator Audit Server records **service access events** and the Cloudera Navigator Metadata Server provides an audit UI and API to view, filter, and export both service access events and the lifecycle and security events retrieved from Cloudera Manager.

Viewing Audit Events

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Audits** tab. The Audit Events report displays all audit events that occurred during the last hour.

Filtering Audit Events

You filter audit events by specifying a time range or adding one or more filters containing an audit event field, operator, and value.

Specifying a Time Range

1. Click the date-time range at the top right of the Audits tab.
2. Do one of the following:

   - Click a **Last** *n* **hours** link.
   - Specify a custom range:

     1. Click **Custom range**.
     2. In the Selected Range endpoints, click each endpoint and specify a date and time in the date control fields.

        - Date - Click the down arrow ▼ to display a calendar and select a date, or click a field and click the spinner arrows ⬍ or up and down arrow keys.
        - Time - Click the hour, minute, and AM/PM fields and click the spinner arrows ⬍ or up and down arrow keys to specify the value.
        - Move between fields by clicking fields or by using the right and left arrow keys.

3. Click **Apply**.

Adding a Filter

1. Do one of the following:

   - Click the ▼ icon that displays next to a field when you hover in one of the event entries.
   - Click the **Filters** link. The Filters pane displays.

     1. Click **Add New Filter** to add a filter.
     2. Choose a field in the **Select Property...** drop-down list. You can search by fields such as username, service name, or operation. The fields vary depending on the service or role. The service name of the Navigator Metadata Server is Navigator.
     3. Choose an operator in the operator drop-down list.
     4. Type a field value in the value text field. To match a substring, use the `like` operator. For example, to see all the audit events for files created in the folder `/user/joe/out`, specify `Source like /user/joe/out`.

   A filter control with field, operation, and value fields is added to the list of filters.

2. Click **Apply**. A field, operation, and value breadcrumb is added above the list of audit events and the list of events displays all events that match the filter criteria.

Removing a Filter

1. Do one of the following:

- Click the **x** next to the filter above the list of events. The list of events displays all events that match the filter criteria.
- Click the **Filters** link. The Filters pane displays.

    1. Click the ▬ at the right of the filter.
    2. Click **Apply**. The filter is removed from above the list of audit event and the list of events displays all events that match the filter criteria.

Service Audit Event Fields

The following fields can appear in a service audit event:

Display Name	Field	Description
Additional Info	additional_info	JSON text that contains more details about an operation performed on entities in Navigator Metadata Server.
Allowed	allowed	Indicates whether the request to perform an operation failed or succeeded. A failure occurs if the user is not authorized to perform the action.
Collection Name	collection_name	The name of the affected Solr collection.
Database Name	database_name	For Sentry, Hive, and Impala, the name of the database on which the operation was performed.
Delegation Token ID	delegation_token_id	Delegation token identifier generated by HDFS NameNode that is then used by clients when submitting a job to JobTracker.
Destination	dest	Path of the final location of an HDFS file in a rename or move operation.
Entity ID	entity_id	Identifier of a Navigator Metadata Server entity. The ID can be retrieved using the Navigator Metadata Server API.
Event Time	timestamp	Date and time an action was performed. The Navigator Audit Server stores the timestamp in the timezone of the Navigator Audit Server. The Navigator UI displays the timestamp converted to the local timezone. Exported audit events contain the stored timestamp.
Family	family	HBase column family.
Impersonator	impersonator	If an action was requested by another service, the name of the user that invoked the action on behalf of the user. • When Sentry is enabled, the Impersonator field displays for services other than Hive. • When Sentry is not enabled, the Impersonator field always displays.
IP Address	ipAddress	The IP address of the host where an action occurred.
Object Type	object_type	For Sentry, Hive, and Impala, the type of the object (TABLE, VIEW, DATABASE) on which operation was performed.
Operation	command	The action performed. • **HBase** - createTable, deleteTable, modifyTable, addColumn, modifyColumn, deleteColumn, enableTable, disableTable, move, assign, unassign, balance, balanceSwitch, shutdown, stopMaster,

Display Name	Field	Description
		flush, split, compact, compactSelection, getClosestRowBefore, get, exists, put, delete, checkAndPut, checkAndDelete, incrementColumnValue, append, increment, scannerOpen, grant, revoke • **HDFS** - setPermission, setOwner, open, concat, setTimes, createSymlink, setReplication, create, append, rename, delete, getfileinfo, mkdirs, listStatus, fsck, listSnapshottableDirectory, setPermission, setReplication • **Hive** - EXPLAIN, LOAD, EXPORT, IMPORT, CREATEDATABASE, DROPDATABASE, SWITCHDATABASE, DROPTABLE, DESCTABLE, DESCFUNCTION, MSCK, ALTERTABLE_ADDCOLS, ALTERTABLE_REPLACECOLS, ALTERTABLE_RENAMECOL, ALTERTABLE_RENAMEPART, ALTERTABLE_RENAME, ALTERTABLE_DROPPARTS, ALTERTABLE_ADDPARTS, ALTERTABLE_TOUCH, ALTERTABLE_ARCHIVE, ALTERTABLE_UNARCHIVE, ALTERTABLE_PROPERTIES, ALTERTABLE_SERIALIZER, ALTERPARTITION_SERIALIZER, ALTERTABLE_SERDEPROPERTIES, ALTERPARTITION_SERDEPROPERTIES, ALTERTABLE_CLUSTER_SORT, SHOWDATABASES, SHOWTABLES, SHOW_TABLESTATUS, SHOW_TBLPROPERTIES, SHOWFUNCTIONS, SHOWINDEXES, SHOWPARTITIONS, SHOWLOCKS, CREATEFUNCTION, DROPFUNCTION, CREATEVIEW, DROPVIEW, CREATEINDEX, DROPINDEX, ALTERINDEX_REBUILD, ALTERVIEW_PROPERTIES, LOCKTABLE, UNLOCKTABLE, ALTERTABLE_PROTECTMODE, ALTERPARTITION_PROTECTMODE, ALTERTABLE_FILEFORMAT, ALTERPARTITION_FILEFORMAT, ALTERTABLE_LOCATION, ALTERPARTITION_LOCATION, CREATETABLE, CREATETABLE_AS_SELECT, QUERY, ALTERINDEX_PROPS, ALTERDATABASE, DESCDATABASE, ALTER_TABLE_MERGE, ALTER_PARTITION_MERGE, GRANT_PRIVILEGE, REVOKE_PRIVILEGE, SHOW_GRANT, GRANT_ROLE, REVOKE_ROLE, SHOW_ROLE_GRANT, CREATEROLE, DROPROLE • **Hue** - USER_LOGIN, USER_LOGOUT, EDIT_USER, ADD_LDAP_USERS, ADD_LDAP_GROUPS, SYNC_LDAP_USERS_GROUPS, EDIT_GROUP, EDIT_PERMISSION, CREATE_USER, CREATE_GROUP, DELETE_USER, DELETE_GROUP • **Impala** - Query, Insert, Update, Delete, GRANT_PRIVILEGE, REVOKE_PRIVILEGE, SHOW_GRANT, GRANT_ROLE, REVOKE_ROLE, SHOW_ROLE_GRANT, CREATEROLE, DROPROLE, DML (Data Manipulation Language statements) • **Navigator Metadata Server** - auditReport, authorization, metadata, policy, search, savedSearch. For the operation subtypes, see Sub Operation. • **Sentry** - GRANT_PRIVILEGE, REVOKE_PRIVILEGE, ADD_ROLE_TO_GROUP, DELETE_ROLE_FROM_GROUP, CREATE_ROLE, DROP_ROLE • **Solr** - add, commit, deleteById, deleteByQuery, finish, query, rollback, CREATE, CREATEALIAS, CREATESHARD, DELETE, DELETEALIAS, DELETESHARD, LIST, LOAD, LOAD_ON_STARTUP, MERGEINDEXES, PERSIST, PREPRECOVERY, RELOAD, RENAME, REQUESTAPPLYUPDATES, REQUESTRECOVERY,

Display Name	Field	Description
		REQUESTSYNCSHARD, SPLIT, SPLITSHARD, STATUS, SWAP, SYNCSHARD, TRANSIENT, UNLOAD
Operation Params	operation_params	Solr query or update parameters used when performing the action.
Operation Text	operation_text	For Sentry, Hive, and Impala, the SQL query that was executed by user. For Hue, the user or group that was added, edited, or deleted.
Permissions	permissions	HDFS permission of the file or directory on which the HDFS operation was performed.
Privilege	privilege	Privilege needed to perform an Impala operation.
Qualifier	qualifier	HBase column qualifier.
Query ID	query_id	The query ID for an Impala operation.
Resource	resource	A service-dependent combination of multiple fields generated during fetch. This field is not supported for filtering as it is not persisted.
Resource Path	resource_path	HDFS URL of Hive objects (TABLE, VIEW, DATABASE, and so on)
Service Name	service	The name of the service that performed the action.
Session ID	session_id	Impala session ID.
Solr Version	solr_version	Solr version number.
Source	src	Path of the HDFS file or directory present in an HDFS operation.
Status	status	Status of an Impala operation providing more information on success or failure.
Stored Object Name	stored_object_name	Name of a policy, saved search, or audit report in Navigator Metadata Server.
Sub Operation	sub_operation	Subtype of operation performed in Navigator Metadata Server. Valid values are: • auditReport - fetchAllReports, fetchAuditReport, createAuditReport, deleteAuditReport, updateAuditReport • authorization - searchGroup, deleteGroup, fetchGroup, fetchRoles, updateRoles • metadata - updateMetadata, fetchMetadata, fetchAllMetadata • policy - fetchAllPolicies, createPolicy, deletePolicy, updatePolicy, fetchPolicySchedule, updatePolicySchedule, deletePolicySchedule • savedSearch - fetchAllSavedSearches, fetchSavedSearch, createSavedSearch, deleteSavedSearch, updateSavedSearch
Table Name	table_name	For Sentry, HBase, Hive, and Impala, the name of the table on which action was performed.
Username	username	The name of the user that performed the action.

Cloudera Navigator Audit Event Reports

Minimum Required Role: Auditing Viewer (also provided by **Full Administrator**)

An *audit report* is a collection of audit events that result from the application of filters. Audit report metadata is recorded by the Cloudera Navigator Metadata Server.

Creating Audit Event Reports

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Audits** tab. The Audit Events report displays all audit events that occurred during the last hour.
3. Do one of the following:

    - Save a filtered version of the Audit Events report:

        1. Optionally specify filters.
        2. Click **Save As Report**.

    - Create a new report by clicking **Create New Report**.

4. Enter a report name.
5. In the **Default time range** field, specify a relative time range. If you had specified a custom absolute time range before selecting **Save As Report**, the *custom absolute time range is discarded*.
6. Optionally add filters.
7. Click **Save**.

Editing Audit Event Reports

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Audits** tab. The Audit Events report displays all audit events that occurred during the last hour.
3. In the left pane, click a report name.
4. Click **Edit Report**.
5. In the **Default time range** field, specify a relative time range. If you had specified a custom absolute time range before selecting **Save As Report**, the *custom absolute time range is discarded*.
6. Optionally add filters.
7. Click **Save**.

Downloading Audit Event Reports

You can download audit event reports in the Navigator UI or by using the Audit API in CSV and JSON formats. An audit event contains the following fields:

- `timestamp`
- `service`
- `username`
- `ipAddress`
- `command`
- `resource`
- `allowed`
- `[operationText]`
- `serviceValues`

Cloudera Navigator Auditing Architecture

The contents of the `resource` and `serviceValues` fields depends on the type of the service. In addition, Hive, Hue, Impala, and Sentry events have the `operationText` field, which contains the operation string. See Service Audit Event Fields on page 38.

In addition to downloading audit events, you can configure the Navigator Audit Server to publish audit events to a Kafka topic or syslog. See Publishing Audit Events.

Downloading Audit Event Reports Using the Audit UI

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Audits** tab. The Audit Events report displays all audit events that occurred during the last hour.
3. Do one of the following:

   - Add filters.
   - In the left pane, click a report name.

4. Select **Export** > *format*, where *format* is CSV or JSON.

Downloading Audit Events Using the Audit API

You can filter and download audit events using the Cloudera Navigator Data Management API.

Hive Audit Events Using the Audit API

To use the API to download the audits events for a service named `hive`, issue the request

```
curl
http://Navigator_Metadata_Server_host:port/api/v9/audits/?query=service%3D%3Dhive&startTime=1431025200000&endTime=1431032400000\
&limit=5&offset=0&format=JSON&attachment=false -X GET -u username:password
```

The `startTime` and `endTime` parameters are required and must be specified in epoch time in milliseconds.

The request could return the following JSON items:

```
[ {
  "timestamp" : "2015-05-07T20:34:39.923Z",
  "service" : "hive",
  "username" : "hdfs",
  "ipAddress" : "12.20.199.170",
  "command" : "QUERY",
  "resource" : "default:sample_08",
  "operationText" : "INSERT OVERWRITE \n  TABLE sample_09 \nSELECT \n
sample_07.code,sample_08.description \n  FROM sample_07 \n  JOIN sample_08 \n  WHERE
sample_08.code = sample_07.code",
  "allowed" : true,
  "serviceValues" : {
    "object_type" : "TABLE",
    "database_name" : "default",
    "operation_text" : "INSERT OVERWRITE \n  TABLE sample_09 \nSELECT \n
sample_07.code,sample_08.description \n  FROM sample_07 \n  JOIN sample_08 \n  WHERE
sample_08.code = sample_07.code",
    "resource_path" : "/user/hive/warehouse/sample_08",
    "table_name" : "sample_08"
  }
}, {
  "timestamp" : "2015-05-07T20:33:50.287Z",
  "service" : "hive",
  "username" : "hdfs",
  "ipAddress" : "12.20.199.170",
  "command" : "SWITCHDATABASE",
  "resource" : "default:",
  "operationText" : "USE default",
  "allowed" : true,
  "serviceValues" : {
    "object_type" : "DATABASE",
    "database_name" : "default",
    "operation_text" : "USE default",
    "resource_path" : "/user/hive/warehouse",
    "table_name" : ""
```

```
      }
   }, {
     "timestamp" : "2015-05-07T20:33:23.792Z",
     "service" : "hive",
     "username" : "hdfs",
     "ipAddress" : "12.20.199.170",
     "command" : "CREATETABLE",
     "resource" : "default:",
     "operationText" : "CREATE TABLE sample_09 (code string,description string) ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '\\t' STORED AS TextFile",
     "allowed" : true,
     "serviceValues" : {
       "object_type" : "DATABASE",
       "database_name" : "default",
       "operation_text" : "CREATE TABLE sample_09 (code string,description string) ROW
   FORMAT DELIMITED FIELDS TERMINATED BY '\\t' STORED AS TextFile",
       "resource_path" : "/user/hive/warehouse",
       "table_name" : ""
     }
   } ]
```

Downloading HDFS Directory Access Permission Reports

Minimum Required Role: Cluster Administrator (also provided by **Full Administrator**)

For each HDFS service, you can download a report that details the HDFS directories a group has permission to access.

1. In the Cloudera Manager Admin Console, click **Clusters** > *ClusterName* > **Reports**.
2. In the Directory Access by Group row, click **CSV** or **XLS**. The Download User Access Report pop-up displays.

    a. In the pop-up, type a group and directory.
    b. Click **Download**. A report of the selected type will be generated containing the following information – path, owner, permissions, and size – for each directory contained in the specified directory that the specified group has access to.

Cloudera Navigator Auditing Use Cases

The Navigator Audit Server tracks the actions performed on the data in a Hadoop cluster. By applying filters on these actions, you can use Cloudera Navigator auditing to view specific information and answer a variety of questions about data and user actions; for example:

- What was a specific user doing on a specific day?
- Who deleted a particular directory?
- What happened to data in a production database, and why is it no longer available?

To answer these questions using Navigator auditing, you begin by logging into the Cloudera Navigator data management UI and clicking the **Audits** tab. Cloudera Navigator displays a list of all audit events for the last hour. The following use cases describe how Navigator can answer some specific questions about data and users.

What Did a User Do on a Specific Day?

In some cases, you may want to identify actions that a specific user performed during a period of time. To determine a user's actions for a time period, you use filters to first specify the user and then define the time period.

The following example identifies the actions of the user named **navigator_user** on June 9, 2016:

1. Filter the list of events for a specific user:

    a. Click **Filters**.
    b. Select **Select Property...** > **Username**.

c. In the field to the right of =, type the username and click **Apply**. The username filter is added to the list of filters, and the list of events is filtered and reloaded. This filter specifies the user **navigator_user**.

Filters ▾

| Username | ⇕ | = | ⇕ | navigator_user | — |

2. Filter the list of events for a specific date and time:

a. Click the date-time field at the top right of the Audit Events page. A set of links display with relative time periods (**Last hour**, **Last 2 hours**, and so on) and a **Custom Range** link that you can use to specify an absolute time range. The Selected Range field displays the currently selected range, which by default is the last hour of the current day.

b. To choose a specific day, click **Custom Range**. The Selected Range field is enabled for input.

c. Use the field controls to choose specific dates and times. The following figure shows the selections for June 9, 2016 12:00 a/m/ to June 10, 2016 12:00 a.m.

| Selected Range: | 06/09/2016, 12:00 AM | - | 06/10/2016, 12:00 AM ⊗ ⇕ ▼ |

d. Click **Apply**.

The following figure shows the first page of the filter results: audit events for the user **navigator_user** during the 24 hour period from June 9, 2016 12:00 a.m. to June 10, 2016 12:00 a.m.

Audit Events
Save As Report

Filters ▾   Username = navigator_user ✕   Jun 9 2016 12:00 AM - Jun 10 2016 12:00 AM ▾

⚙ Export ▾   ‹ 1 - 50 ›

▸	Timestamp	Username	IP Address	Service Name	Operation	Resource
▸	Jun 9 2016 7:46 PM	navigator_user ▾	10.17.207.26 ▾	hdfs ▾	delete ▾	/user/hive/warehouse/sample_09/.hive-staging_hive_2016-06
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	listStatus	/user/hive/warehouse
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	getfileinfo	/user/hive/warehouse
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	listStatus	/user/hive
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	getfileinfo	/user/hive
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	getfileinfo	/
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	listStatus	/user
▸	Jun 9 2016 2:40 PM	navigator_user	10.17.207.29	hdfs	getfileinfo	/user

Who Deleted Files from the Hive Warehouse Directory?

The Hive warehouse directory is usually set to `/user/hive/warehouse`. In this example, files have been deleted from the directory and you want to identify who removed them.

To determine who deleted files from this directory, use filters in Cloudera Manager to do the following:

1. Filter the list of events for the source `/user/hive/warehouse`:

a. Click **Filters**.

b. Select **Select Property...** > **Source**.

c. In the operator field, select **like**.

d. In the empty field to the right of **like**, type `/user/hive/warehouse` and click **Apply**. The source filter is added to the list of filters and the list of events is filtered and reloaded.

2. Filter the list of events for the delete operation:

a. Click **Add New Filter**.

b. Select **Select Property...** > **Operation**.

c. In the operator field, select **=**.

d. In the empty field to the right of **=**, type `delete` and click **Apply**. The operation filter is added to the list of filters and the list of events is filtered and reloaded.

The following figure shows the resulting filters.

The following figure shows the results of the filters: **navigator_user** deleted or attempted to delete (indicated by the red text) the displayed resources from the Hive warehouse directory during the 30-day period from May 28, 2016 to June 27, 2016.

What Happened to Data in the Database?

Typically, data in the database is partitioned into folders or files labeled by date. In this example, data from 2015 is missing from the production database, and you want to find out what happened to it. You can use Cloudera Navigator to determine what happened to data that was created during this period of time.

Data created in 2015 has the string "2015" in the filename. To determine what happened to the data stored in folders and files in the year 2015, do the following:

1. Filter the list of events for sources containing the string "2015":

    a. Click **Filters**.
    b. Select **Select Property...** > **Source** to specify the path of an HDFS file or directory.
    c. In the operator field, select **like**.
    d. In the empty field to the right of **like**, type `2015` and click **Apply**. The source filter is added to the list of filters, and the list of events is filtered and reloaded.

2. Filter the list of events for the delete operation:

    a. Click **Add New Filter**.
    b. Select **Select Property...** > **Operation**.
    c. In the operator field, select **=**.
    d. In the empty field to the right of **=**, type `delete` and click **Apply**. The operation filter is added to the list of filters and the list of events is filtered and reloaded.

3. Set the date range to one year:

    a. Click the date-time field at the top right of the Audit Events page.
    b. To set the range to be the last year, click **Custom Range**. The Selected Range field is enabled for input.
    c. In the left date field, use the field controls to specify a date one year ago.
    d. Click **Apply**.

The following figure shows the resulting filters.

The following figure shows the results of the filter application. During the last year, the user **hdfs** deleted the directories with names that contain "2015":

Cloudera Navigator Analytics

Cloudera Navigator allows you to view metadata and audit analytics for HDFS entities. On the analytics pages, you can view which HDFS entities satisfy the following property values:

- Metadata - the number of files by creation and access times, size, block size, and replication count. After selecting a property value range for one of these properties, you can filter the matching files by directory, owner, and tag.
- Audit
    - Activity tab - by directory which files have been accessed using the open operation and how many times they have been accessed. Activity analytics are based on summarized data computed once a day and will not match the number of events viewed in the Audits tab at all times.
    - Top Users tab - the top-*n* commands and the top-*n* users and top *n* commands those users performed during various time windows (1 min–1 day), where *n* is 1, 5, 10, 20, 50, or 100.

Viewing Metadata Analytics

Minimum Required Role: Lineage Viewer and **Policy Administrator** (also provided by **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Analytics** tab. The Metadata analytics tab displays.
3. Choose an HDFS service instance from the *service_name* Analytics drop-down list.
4. The Metadata tab displays a set of bar graphs that list the number of files that satisfy groups of values for last access time, created time, size, block size, and replication count.

    - To display the files at the right, click a bar. This draws a blue selection outline around the bar and selects the property checkbox.
    - To select more than one value, grab a bar edge and brush a range of values.
    - To change a range, click a bar, drag to a different range of values, and drop.
    - To reduce a range, grab a bar edge and contract the range.
    - To clear a property, clear the checkbox. The previous selection is indicated with a gray outline.
    - When you select a previously selected property, the previous selection is reused. For example, if you had previously selected one and three for replication count, and you reselect the replication count checkbox, the values one and three are reselected.
    - To clear all selections, present and previous, click **Clear all selections**.

5. In the listing on the right, select an option to display the number of files by directory, owner, or tag. In the listing:

    - Filter the selections by typing strings in the search box and pressing **Enter** or **Return**.
   - Add categories (directory, owner, or tag) to a search query and display the Search tab by doing one of the following:
        - Clicking a directory, owner, or tag name link.
        - Selecting **Actions** > **Show in search**. To further refine the query, select one or more checkboxes, and select **Actions** > **Show selection in search**.
    - **Minimum Required Role: Policy Administrator** (also provided by **Full Administrator**)

      Add categories to the search query of a new policy and display the Policies tab by selecting **Actions** > **Create a policy**. To further refine the query, select one or more checkboxes, and select **Actions** > **Create a policy from selection**.

Viewing Audit Analytics

Minimum Required Role: Auditing Viewer (also provided by **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.

2. Click the **Analytics** tab. If the logged-in user has a role that permits access to metadata analytics, the Metadata analytics tab displays.

3. Choose an HDFS service instance from the *service_name* Analytics drop-down list.

4. If not already displayed, click the **Audits** tab. The Activity tab displays a bar graph that lists the number of files that have been read the number of times listed in the x-axis.

   - To display at the right the directories containing the files that have been read, click an activity bar. This draws a blue selection outline around the bar and selects the Activity checkbox.
   - To select more than one value, grab a bar edge and brush a range of values.
   - To change a range, click a bar, drag to a different range of values, and drop.
   - To reduce a range, grab a bar edge and contract the range.
   - To clear Activity, clear the checkbox. The previous selection is indicated with a gray outline.
   - When you select Activity and the graph had a previous selection, the previous selection is reused. For example, if you had previously selected values spanning six through nine for the number of times files have been read, and you select the checkbox, six through nine will be reselected.

5. In the directory listing on the right:

   - Filter the directories by typing directory strings in the search box and pressing **Enter** or **Return**.
   - **Minimum Required Role: Lineage Administrator** (also provided by **Metadata Administrator**, **Full Administrator**)

     Add selected directories to a search query and display the Search tab by doing one of the following:

     – Clicking a directory name link.
     – Selecting one or more directory checkboxes and selecting **Actions** > **Show selection in search**.

   - **Minimum Required Role: Metadata Viewer** (also provided by **Metadata Administrator, Full Administrator**)

     Minimum Required Role: Lineage Administrator (also provided by **Metadata Administrator**, **Full Administrator**)

     Add selected directories to the search query of a new policy and display the Policies tab by selecting one or more directory checkboxes and selecting **Actions** > **Create a policy from selection**.

For example, the following screenshot shows files that have been accessed two and three times, match the string `sample`, and are in the `/usr/hive/warehouse/sample*` directories. Each directory has one file that has been accessed.
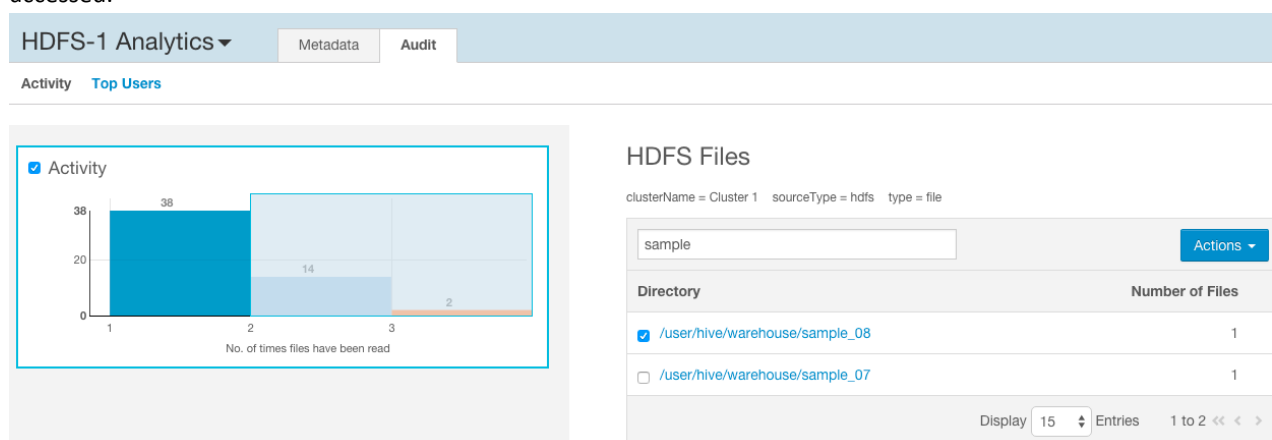
Metadata Policies

A ***metadata policy*** defines a set of actions performed by the Cloudera Navigator Metadata Server on a class of entities. You can perform the following actions:

- Add custom metadata such as tags and properties.
- Run a command, such as moving an HDFS entity to another location or moving an HDFS entity to HDFS trash.

  If a policy creator configures a command action to move a directory and the creator does not have access to the directory, the action fails. Similarly, if a policy creator does not have access to a file in the directory, the action fails. To ensure that command actions do not fail, policies containing command actions should be created by data stewards, who are members of a user group that has the appropriate access to HDFS files.

- Send a message to a JMS message queue. The JSON format message contains the metadata of the entity to which the policy applies and the message text specified in the policy:

```
{"entity":entity_properties, "userMessage":"some message text"}
```

  To send a message to a JMS message queue, you must configure the JMS server properties.

For some actions, certain properties support specifying a value using a policy expression.

A policy is run as the user who created the policy, in the home directory of the user who created the policy. To change who a policy runs as, log into Navigator as the new user you want to run the policy as, clone the policy as the new user, and then delete or disable the old policy.

Viewing Policies

Minimum Required Role: Policy Viewer (also provided by **Policy Administrator**, **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Policies** tab.

Viewing a Policy

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Policies** tab.
3. In a policy row, click a policy name link or select **Actions** > **View**.

Enabling and Disabling Policies

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Policies** tab.
3. In a policy row, click a policy name link or select **Actions** > **Enable** or **Actions** > **Disable**.

Creating Policies

Minimum Required Role: Policy Administrator (also provided by **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.
2. Depending on the starting point, do one of the following:

Action	Procedure
Policies page	1. Click the **Policies** tab. 2. Click **Create New Policy**.
Search results page	1. Select **Actions** > **Create a policy**.

3. In the Status field, check the **Enable** checkbox.
4. Enter a name for the policy.
5. Specify the search query that defines the class of entities to which the policy applies. If you arrive at the Policies page by clicking a search result, the query property is populated with the query that generated the result. To display a list of entities that satisfy a search query, click the **Search Results** link.
6. Specify an optional description for the policy.
7. If you choose to use policy expressions in properties that support expressions, specify required imports in the **Import Statements** field. See Metadata Policy Expression Examples on page 51.
8. Choose the schedule for applying the policy:

   - **On Change** - When the entities matching the search string change.
   - **Immediate** - When the policy is created.
   - **Once** - At the time specified in the Start Time field.
   - **Recurring** - At recurring times specified by the Start and End Time fields at the interval specified in the Interval field.

   For the Once and Recurring fields, specify dates and times as follows:

   - Date - Click the down arrow ▼ to display a calendar and select a date, or click a field and click the spinner arrows ▲▼ or up and down arrow keys.

   - Time - Click the hour, minute, and AM/PM fields and click the spinner arrows ▲▼ or up and down arrow keys to specify the value.
   - Move between fields by clicking fields or by using the right and left arrow keys.

9. Follow the appropriate procedure for the actions performed by the policy.

Action	Procedure
Assign Metadata	1. Specify the custom metadata or managed metadata to be assigned. Only managed metadata of type Text and single valued is supported. Optionally, check the **Expression** checkbox and specify a policy expression for the fields that support expressions.
Configure Command Actions	1. Select **Add Action** > **Move to Trash** or **Add Action** > **Move**. For a move, specify the location to move the entity to in the Target Path field. If you specify multiple actions, they are run in the order in which they are specified. Command actions are supported only for HDFS entities. If you configure a command action for unsupported entities, a runtime error will be logged when the policy runs. See Viewing Command Action Status on page 32.
Send Notification to JMS	1. If not already configured, configure a JMS server and queue. 2. Specify the queue name and message. Optionally, check the **Expression** checkbox and specify a policy expression for the message.

10. Click **Save**.

Copying and Editing a Policy

Minimum Required Role: Policy Administrator (also provided by **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Policies** tab.
3. In a policy row, select **Actions** > **Copy** or **Actions** > **Edit**.
4. Edit the policy name, search query, or policy actions.

5. Click **Save**.

Deleting Policies

Minimum Required Role: Policy Administrator (also provided by **Full Administrator**)

1. Start and log into the Cloudera Navigator data management component UI.
2. Click the **Policies** tab.
3. In a policy row, select **Actions** > **Delete** and **OK** to confirm.

Metadata Policy Expressions

A *metadata policy expression* allows you to specify certain metadata extraction policy properties using Java expressions instead of string literals. The supported properties are: entity name and description, key-value pairs, and JMS notification message.

You must declare classes accessed in the expression in the policy's **Import Statements** field. A metadata policy expression must evaluate to a string.

Metadata policy expressions are not enabled by default. To enable metadata policy expressions, follow the procedure in Enabling and Disabling Metadata Policy Expression Input.

Including Entity Properties in Policy Expressions

To include entity properties in property expressions, use the `entity.get` method, which takes a property and a return type:

```
entity.get(XXProperties.Property, return_type)
```

XX`Properties.`*Property* is the Java enumerated value representing an entity property, where

- *XX* is FSEntity, HiveColumn, HiveDatabase, HivePartition, HiveQueryExecution, HiveQueryPart, HiveQuery, HiveTable, HiveView, JobExecution, Job, WorkflowInstance, Workflow, PigField, PigOperationExecution, PigOperation, PigRelation, SqoopExportSubOperation, SqoopImportSubOperation, SqoopOperationExecution, SqoopQueryOperation, SqoopTableExportOperation, or SqoopTableImportOperation.
- *Property* is one of the properties listed in Entity Property Enum Reference on page 52.

If you do not need to specify a return type, use `Object.class` as the return type. However, if you want to do type-specific operations with the result, set the return type to the type in the comment in the enum property reference. For example, in `FSEntityProperties`, the return type of the `ORIGINAL_NAME` property is `java.lang.String`. If you use `String.class` as the return type, you can use the `String` method `toLowerCase()` to modify the returned value: `entity.get(FSEntityProperties.ORIGINAL_NAME, String.class).toLowerCase()`.

Metadata Policy Expression Examples

- Set a filesystem entity name to the original name concatenated with the entity type:

```
entity.get(FSEntityProperties.ORIGINAL_NAME, Object.class) + " " +
entity.get(FSEntityProperties.TYPE, Object.class)
```

  Import Statements:

```
import com.cloudera.nav.hdfs.model.FSEntityProperties;
```

- Add the entity's creation date to the entity name:

```
entity.get(FSEntityProperties.ORIGINAL_NAME, Object.class) + " - "
+ new SimpleDateFormat("yyyy-MM-dd").format(entity.get(FSEntityProperties.CREATED,
Instant.class).toDate())
```

Import Statements:

```
import com.cloudera.nav.hdfs.model.FSEntityProperties; import java.text.SimpleDateFormat;
 import org.joda.time.Instant;
```

- Set the key-value pair: retain_util-seven years from today's local time:

```
new DateTime().plusYears(7).toLocalDateTime().toString("MMM dd yyyy", Locale.US)
```

Import statements:

```
import org.joda.time.DateTime; import java.util.Locale;
```

Entity Property Enum Reference

The following reference lists the Java enumerated values for retrieving properties of each entity type.

```
com.cloudera.nav.hdfs.model.FSEntityProperties
public enum FSEntityProperties implements PropertyEnum {
  PERMISSIONS, // Return type: java.lang.String
  TYPE, // Return type: java.lang.String
  SIZE, // Return type: java.lang.Long
  OWNER, // Return type: java.lang.String
  LAST_MODIFIED, // Return type: org.joda.time.Instant
  SOURCE_TYPE, // Return type: java.lang.String
  DELETED, // Return type: java.lang.Boolean
  FILE_SYSTEM_PATH, // Return type: java.lang.String
  CREATED, // Return type: org.joda.time.Instant
  LAST_ACCESSED, // Return type: org.joda.time.Instant
  GROUP, // Return type: java.lang.String
  MIME_TYPE, // Return type: java.lang.String
  DELETE_TIME, // Return type: java.lang.Long
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveColumnProperties
public enum HiveColumnProperties implements PropertyEnum {
  TYPE, // Return type: java.lang.String
  SOURCE_TYPE, // Return type: java.lang.String
  DELETED, // Return type: java.lang.Boolean
  DATA_TYPE, // Return type: java.lang.String
  ORIGINAL_DESCRIPTION, // Return type: java.lang.String
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveDatabaseProperties
public enum HiveDatabaseProperties implements PropertyEnum {
  TYPE, // Return type: java.lang.String
  ORIGINAL_DESCRIPTION, // Return type: java.lang.String
  SOURCE_TYPE, // Return type: java.lang.String
  DELETED, // Return type: java.lang.Boolean
  FILE_SYSTEM_PATH, // Return type: java.lang.String
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
```

```
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HivePartitionProperties
public enum HivePartitionProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    DELETED, // Return type: java.lang.Boolean
    FILE_SYSTEM_PATH, // Return type: java.lang.String
    CREATED, // Return type: org.joda.time.Instant
    LAST_ACCESSED, // Return type: org.joda.time.Instant
    COL_VALUES, // Return type: java.util.List
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveQueryExecutionProperties
public enum HiveQueryExecutionProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    ENDED, // Return type: org.joda.time.Instant
    INPUTS, // Return type: java.util.Collection
    OUTPUTS, // Return type: java.util.Collection
    STARTED, // Return type: org.joda.time.Instant
    PRINCIPAL, // Return type: java.lang.String
    WF_INST_ID, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveQueryPartProperties
public enum HiveQueryPartProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveQueryProperties
public enum HiveQueryProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    INPUTS, // Return type: java.util.Collection
    OUTPUTS, // Return type: java.util.Collection
    QUERY_TEXT, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    WF_IDS, // Return type: java.util.Collection
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveTableProperties
public enum HiveTableProperties implements PropertyEnum {
```

```
    OWNER, // Return type: java.lang.String
    INPUT_FORMAT, // Return type: java.lang.String
    OUTPUT_FORMAT, // Return type: java.lang.String
    DELETED, // Return type: java.lang.Boolean
    FILE_SYSTEM_PATH, // Return type: java.lang.String
    COMPRESSED, // Return type: java.lang.Boolean
    PARTITION_COL_NAMES, // Return type: java.util.List
    CLUSTERED_BY_COL_NAMES, // Return type: java.util.List
    SORT_BY_COL_NAMES, // Return type: java.util.List
    SER_DE_NAME, // Return type: java.lang.String
    SER_DE_LIB_NAME, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    CREATED, // Return type: org.joda.time.Instant
    LAST_ACCESSED, // Return type: org.joda.time.Instant
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.hive.model.HiveViewProperties
public enum HiveViewProperties implements PropertyEnum {
    DELETED, // Return type: java.lang.Boolean
    QUERY_TEXT, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    CREATED, // Return type: org.joda.time.Instant
    LAST_ACCESSED, // Return type: org.joda.time.Instant
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.mapreduce.model.JobExecutionProperties
public enum JobExecutionProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    JOB_ID, // Return type: java.lang.String
    ENDED, // Return type: org.joda.time.Instant
    INPUT_RECURSIVE, // Return type: boolean
    TYPE, // Return type: java.lang.String
    INPUTS, // Return type: java.util.Collection
    OUTPUTS, // Return type: java.util.Collection
    STARTED, // Return type: org.joda.time.Instant
    PRINCIPAL, // Return type: java.lang.String
    WF_INST_ID, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.mapreduce.model.JobProperties
public enum JobProperties implements PropertyEnum {
    ORIGINAL_NAME, // Return type: java.lang.String
    INPUT_FORMAT, // Return type: java.lang.String
    OUTPUT_FORMAT, // Return type: java.lang.String
    OUTPUT_KEY, // Return type: java.lang.String
    OUTPUT_VALUE, // Return type: java.lang.String
    MAPPER, // Return type: java.lang.String
    REDUCER, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
```

```
    WF_IDS, // Return type: java.util.Collection
    NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.oozie.model.WorkflowInstanceProperties
public enum WorkflowInstanceProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    CREATED, // Return type: org.joda.time.Instant
    JOB_ID, // Return type: java.lang.String
    STATUS, // Return type: java.lang.String
    ENDED, // Return type: org.joda.time.Instant
    INPUTS, // Return type: java.util.Collection
    OUTPUTS, // Return type: java.util.Collection
    STARTED, // Return type: org.joda.time.Instant
    PRINCIPAL, // Return type: java.lang.String
    WF_INST_ID, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.oozie.model.WorkflowProperties
public enum WorkflowProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    WF_IDS, // Return type: java.util.Collection
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.pig.model.PigFieldProperties
public enum PigFieldProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    INDEX, // Return type: int
    SOURCE_TYPE, // Return type: java.lang.String
    DATA_TYPE, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.pig.model.PigOperationExecutionProperties
public enum PigOperationExecutionProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    ENDED, // Return type: org.joda.time.Instant
    INPUTS, // Return type: java.util.Collection
    OUTPUTS, // Return type: java.util.Collection
    STARTED, // Return type: org.joda.time.Instant
    PRINCIPAL, // Return type: java.lang.String
    WF_INST_ID, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
```

```
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.pig.model.PigOperationProperties
public enum PigOperationProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    OPERATION_TYPE, // Return type: java.lang.String
    SCRIPT_ID, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    WF_IDS, // Return type: java.util.Collection
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.pig.model.PigRelationProperties
public enum PigRelationProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    FILE_SYSTEM_PATH, // Return type: java.lang.String
    SCRIPT_ID, // Return type: java.lang.String
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopExportSubOperationProperties
public enum SqoopExportSubOperationProperties implements PropertyEnum {
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    INPUTS, // Return type: java.util.Collection
    FIELD_INDEX, // Return type: int
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopImportSubOperationProperties
public enum SqoopImportSubOperationProperties implements PropertyEnum {
    DB_COLUMN_EXPRESSION, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    SOURCE_TYPE, // Return type: java.lang.String
    INPUTS, // Return type: java.util.Collection
    FIELD_INDEX, // Return type: int
    NAME, // Return type: java.lang.String
    ORIGINAL_NAME, // Return type: java.lang.String
    USER_ENTITY, // Return type: boolean
    SOURCE_ID, // Return type: java.lang.String
    EXTRACTOR_RUN_ID, // Return type: java.lang.String
    PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopOperationExecutionProperties
public enum SqoopOperationExecutionProperties implements PropertyEnum {
    SOURCE_TYPE, // Return type: java.lang.String
    TYPE, // Return type: java.lang.String
    ENDED, // Return type: org.joda.time.Instant
```

```
  INPUTS, // Return type: java.util.Collection
  OUTPUTS, // Return type: java.util.Collection
  STARTED, // Return type: org.joda.time.Instant
  PRINCIPAL, // Return type: java.lang.String
  WF_INST_ID, // Return type: java.lang.String
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopQueryOperationProperties
public enum SqoopQueryOperationProperties implements PropertyEnum {
  SOURCE_TYPE, // Return type: java.lang.String
  INPUTS, // Return type: java.util.Collection
  QUERY_TEXT, // Return type: java.lang.String
  DB_USER, // Return type: java.lang.String
  DB_URL, // Return type: java.lang.String
  OPERATION_TYPE, // Return type: java.lang.String
  TYPE, // Return type: java.lang.String
  WF_IDS, // Return type: java.util.Collection
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopTableExportOperationProperties
public enum SqoopTableExportOperationProperties implements PropertyEnum {
  DB_TABLE, // Return type: java.lang.String
  SOURCE_TYPE, // Return type: java.lang.String
  DB_USER, // Return type: java.lang.String
  DB_URL, // Return type: java.lang.String
  OPERATION_TYPE, // Return type: java.lang.String
  TYPE, // Return type: java.lang.String
  WF_IDS, // Return type: java.util.Collection
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

```
com.cloudera.nav.sqoop.model.SqoopTableImportOperationProperties
public enum SqoopTableImportOperationProperties implements PropertyEnum {

  DB_TABLE, // Return type: java.lang.String
  DB_WHERE, // Return type: java.lang.String
  SOURCE_TYPE, // Return type: java.lang.String
  DB_USER, // Return type: java.lang.String
  DB_URL, // Return type: java.lang.String
  OPERATION_TYPE, // Return type: java.lang.String
  TYPE, // Return type: java.lang.String
  WF_IDS, // Return type: java.util.Collection
  NAME, // Return type: java.lang.String
  ORIGINAL_NAME, // Return type: java.lang.String
  USER_ENTITY, // Return type: boolean
  SOURCE_ID, // Return type: java.lang.String
  EXTRACTOR_RUN_ID, // Return type: java.lang.String
  PARENT_PATH; // Return type: java.lang.String
}
```

Cloudera Navigator Lineage Diagrams

Minimum Required Role: [Lineage Administrator](#) (also provided by **Metadata Administrator**, **Full Administrator**)

Cloudera Navigator provides an automatic collection and easy visualization of upstream and downstream data lineage to verify reliability. For each data source, it shows, down to the column level within that data source, what the precise upstream data sources were, the transforms performed to produce it, and the impact that data has on downstream artifacts.

A *lineage diagram* is a directed graph that depicts an extracted entity and its relations with other entities. A lineage diagram is limited to 400 entities. Once that limit is reached, certain entities display as a "hidden" icon.

Entities

In a lineage diagram, entity types are represented by icons:

HDFS		Pig	
• File • Directory	• •	• Table • Pig script • Pig script execution	• • •
Hive and Impala		**Spark** (Unsupported and disabled by default. To enable, see [Enabling Spark Metadata Extraction](#).)	
• Table • Query template • Query execution	• • •	• Job template • Job execution	• •
MapReduce and YARN		**Sqoop**	
• Job template • Job execution	• •	• Job template • Job execution	• •
Oozie		**Hidden**	
• Job template • Job execution	• •	•	See [Viewing the Lineage of Hidden Entities](#) on page 61.

> **Important:** Tables created by Impala queries and Sqoop jobs are represented as Hive entities.

In the following circumstances, the entity type icon appears as

    :
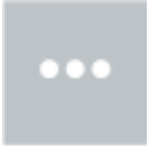
- The entity has not yet been extracted. In this case,

  is eventually replaced with the correct entity icon after the entity is extracted and linked in Navigator. For information on how long it takes for newly created entities to be extracted, see Metadata Extraction on page 9.
- A Hive entity has been deleted from the system before it could be extracted.

Parent entities are represented by a blue box enclosing other entities. The following lineage diagram illustrates the relations between the YARN job `script.pig` and Pig script `script.pig` invoked by the parent Oozie workflow `pig-app-hue-script` and the source file in the `data` folder and destination folder `poutput`:
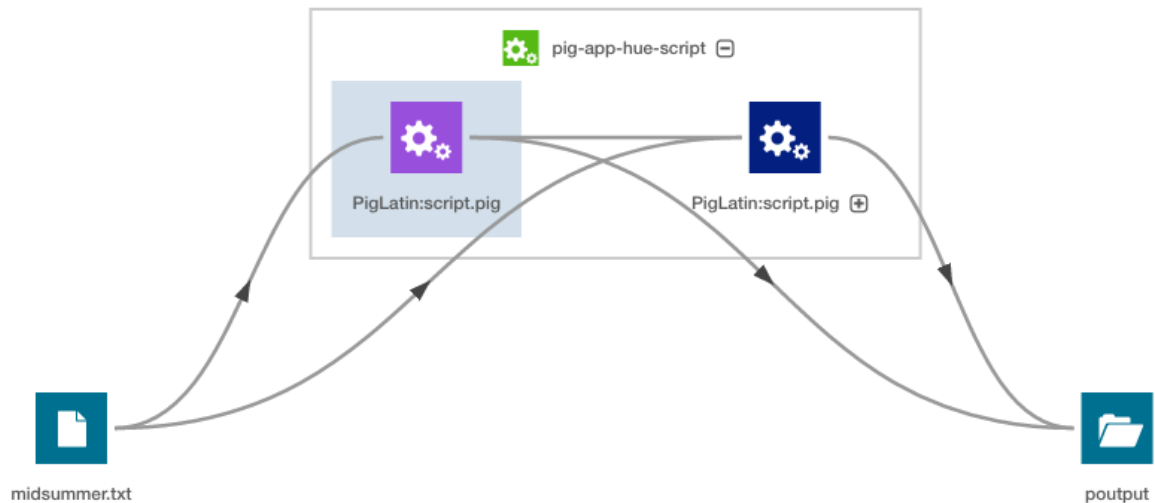
Relations

Relations between the entities are represented graphically by lines, with arrows indicating the direction of the data flow. Navigator supports the following types of relations:

Relation Type	Description
Data flow	Describes a relation between data and a processing activity; for example, between a file and a MapReduce job or vice versa.
Parent-child	Describes a parent-child relation. For example, between a directory and a file.
Logical-physical	Describes the relation between a logical entity and its physical entity. For example, between a Hive query and a MapReduce job.
Instance of	Describes the relation between a template and its instance. For example, an operation execution is an instance of operation. Instance of relations are never visualized in the lineage, however you can navigate between template and instance lineage diagrams. See Displaying an Instance Lineage Diagram on page 67 and Displaying the Template Lineage Diagram for an Instance Lineage Diagram on page 67.

Relation Type	Description
Control flow	Describes a relation where the source entity controls the data flow of the target entity. For example, between the columns used in an `insert` clause and the `where` clause of a Hive query.

Lineage diagrams contain the following line types:

- Solid ( ) represents a "data flow" relationship, indicating that the columns appear (possibly transformed) in the output (when directional with arrow) and "logical- physical" (when no arrow). For example, a solid line appears between the columns used in a `select` clause.

- Dashed ( ) represents a "control flow" relationship, indicating that the columns determine which rows flow to the output. For example, a dashed line appears between the columns used in an `insert` or `select` clause and the `where` clause of a Hive query. Control flow lines are hidden by default. See Filtering Lineage Diagrams on page 62.

- Blue ( ) represents a selected link.

- Green ( ) represents a summary link that contains operations. When you click the link, the link turns blue (for selected) and the nested operations display in the selected link summary:

The following query:

```
SELECT sample_07.description,sample_07.salary FROM sample_07
WHERE ( sample_07.salary > 100000)
ORDER BY sample_07.salary DESC LIMIT 1000
```

has solid, directed lines between the columns in the `select` clause and a dashed line between the columns in the `where` clause:
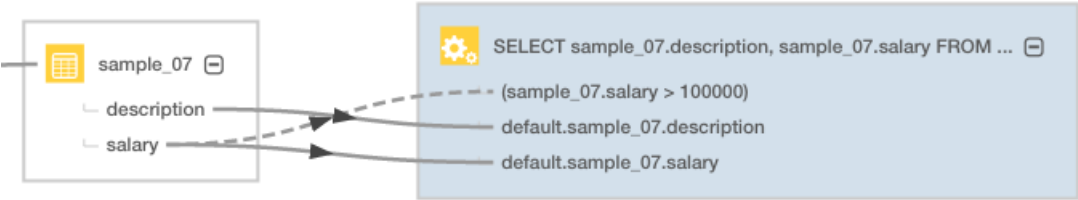
Manipulating Lineage Diagrams

Expanding Entities

You can click a ⊕ icon in a parent entity to display its child entities. For example, you can click an Oozie job to display its child Pig script and the Pig script to display its child tables:
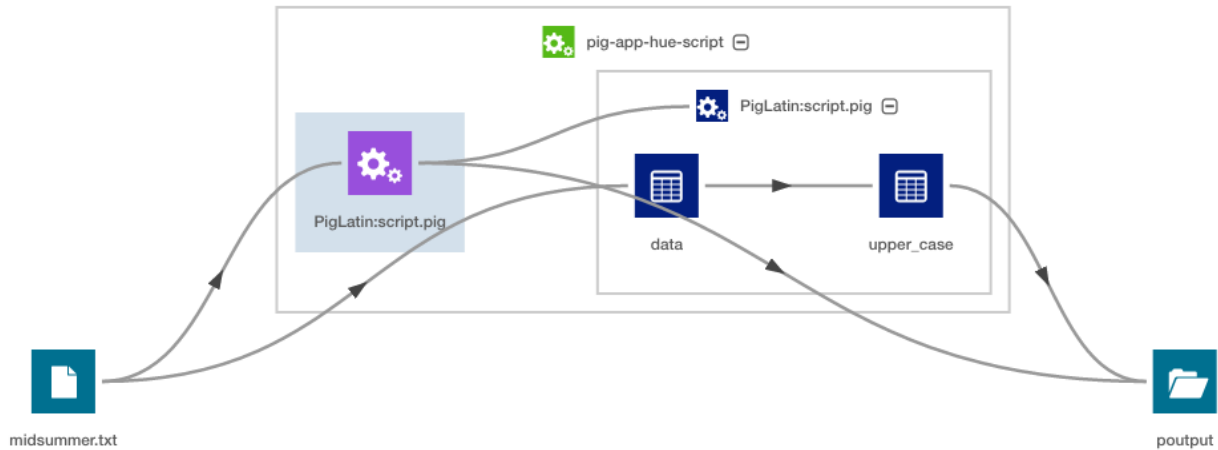
Modifying Lineage Layout

- To improve the layout of a lineage diagram, you can drag entities (in the diagram above `midsummer.txt` and `pout`) located outside a parent box.
- Use the mouse scroll wheel or the

  control to zoom the lineage diagram in and out.
- You can move an entire lineage diagram in the lineage pane by pressing the mouse button and dragging it.

Viewing the Lineage of Hidden Entities

Lineage that is not fully traversed (that is, you do not see a subset of the actual lineage) is illustrated by the

icon. This icon displays when the lineage diagram has more than 400 entities. For example:

One or more links from sample_08 were not included in this lineage. To explore these links further, view the lineage of sample_08
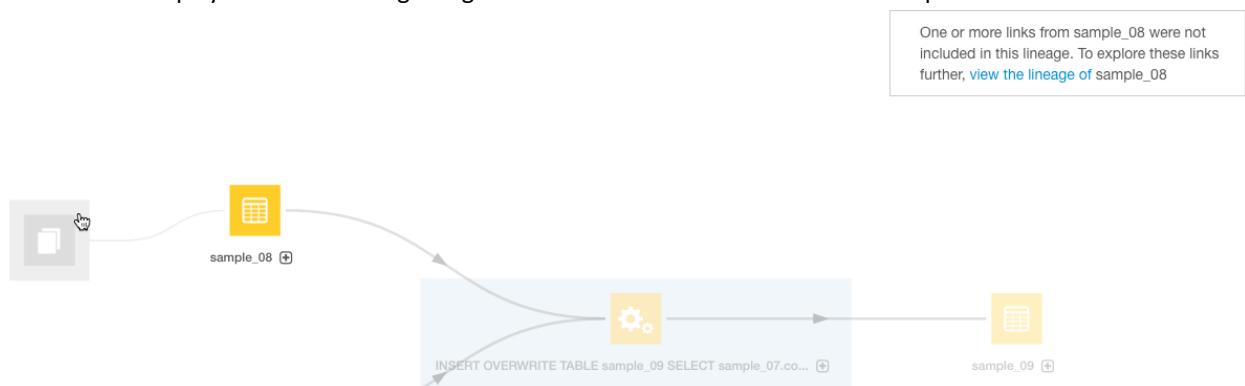
To view the lineage of hidden entities, select the hidden entity and click **view the lineage** in the box on the right to display a new lineage centered around that entity. After clicking the link, you would see the following:

Filtering Lineage Diagrams

To reduce the time and resources required to render large lineage diagrams, you can filter out classes of entities and links by selecting checkboxes in the **Lineage Options** box on the right of the diagram. The following are the default selections:

The **Only Upstream/Downstream** filter allows you to filter out entities and links that are input (upstream) to and output (downstream) from another entity.

Use the **Latest Partition and Operation** filter to reduce rendering time when you have similar partitions created and operations performed periodically. For example, if Hive partitions are created daily, the filter allows you to display only the latest partition.

Filter Example

If you display the lineage of the `sample_09` table with no filtering options selected (other than hiding deleted items), the lineage appears as follows.
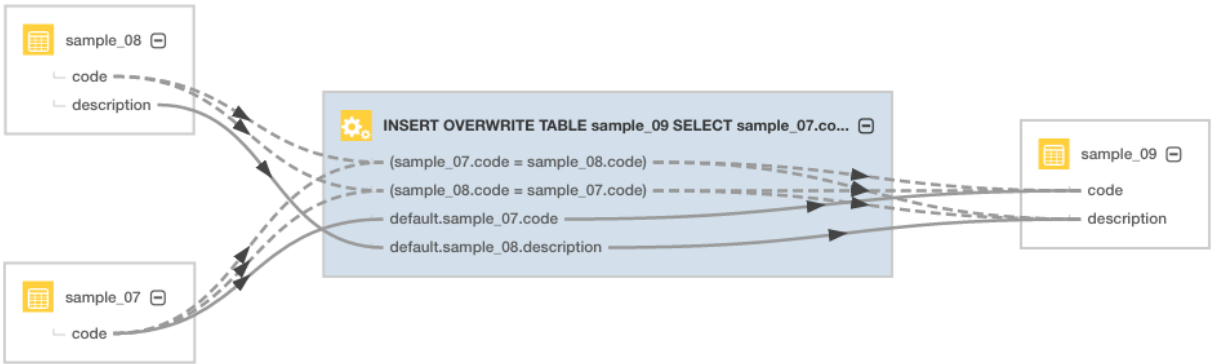
Subsequent diagrams show the result of using each supported filter type:

- **Control Flow Relations** - The operation is collapsed and control flow links are hidden.
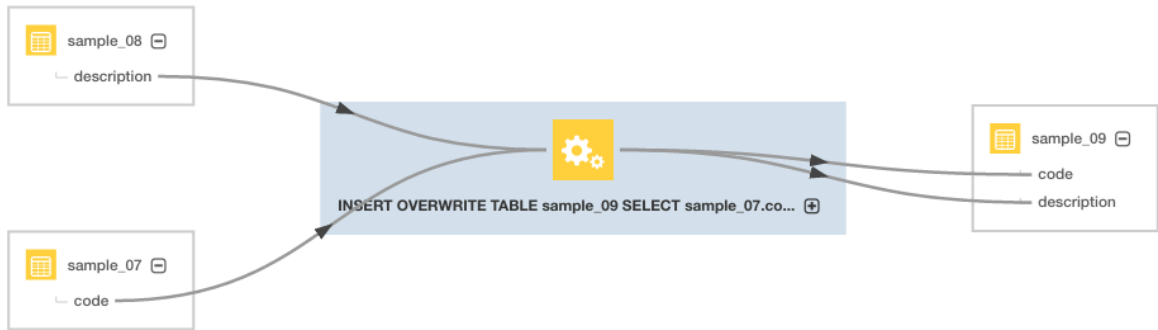
- **Show Upstream** and **Show Downstream** - The operation is collapsed and only upstream entities and links are shown. The output table is hidden.
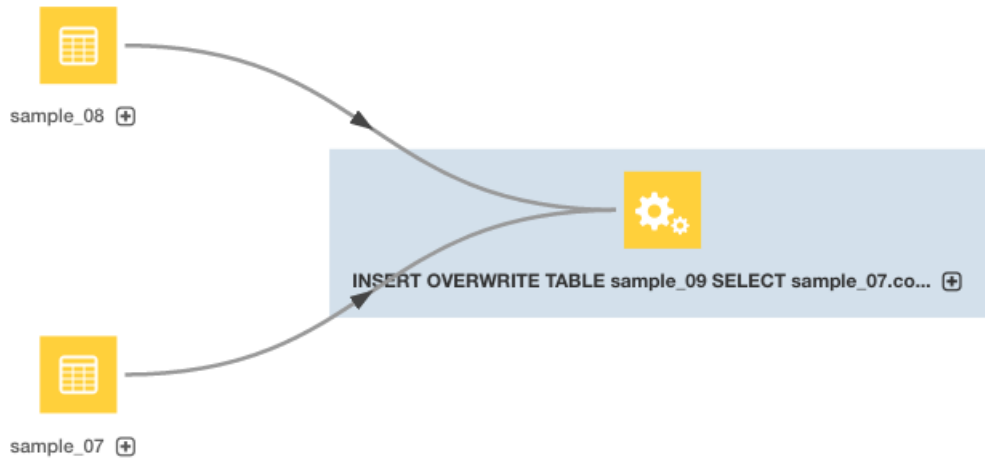
Here, the operation is collapsed and only downstream entities and links are shown. The input tables are hidden.
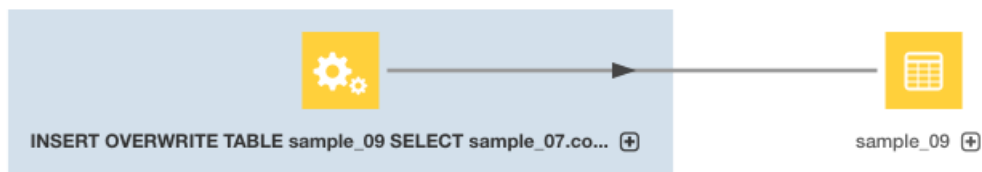
- **Operations** - In the diagram, the operation is hidden.
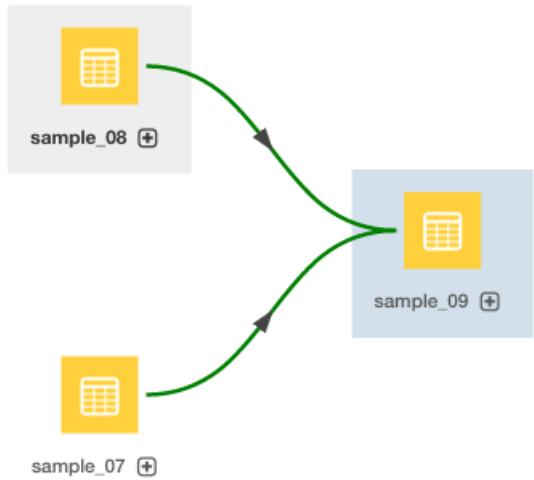
The green links indicate that one or more operations are collapsed into the links.

- **Deleted Entities** - Here, the operation is hidden but deleted entities are displayed.
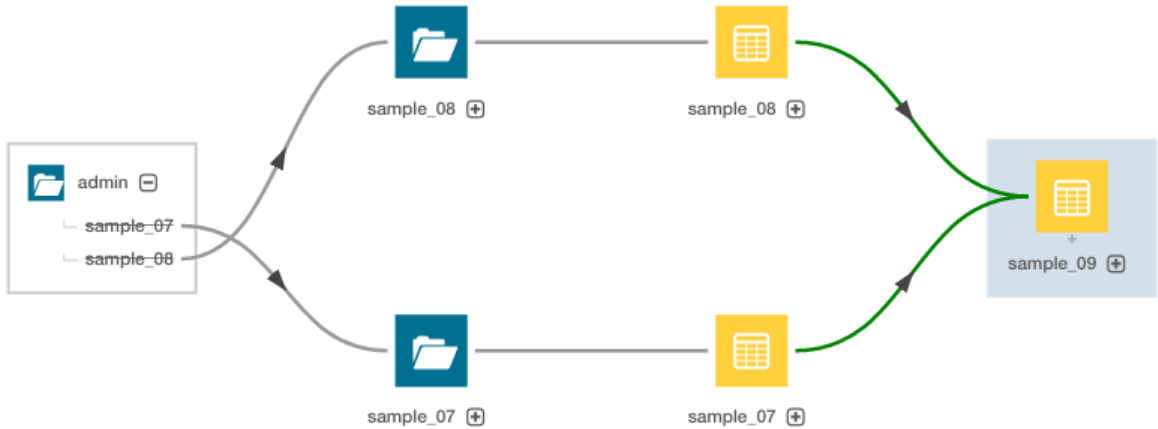
Searching a Diagram

You can search a lineage diagram for an entity by doing the following:

1. In the Search box at the right of the diagram, type an entity name. A list of matching entities displays below the box.
2. Click an entity in the list. A blue box is drawn around the entity and the entity details display in a box below the Search box.
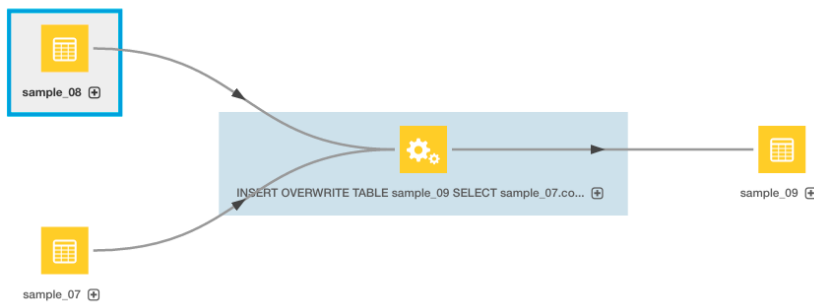
3. Click the **Show** link next to the entity. The selected entity moves to the center of the diagram.
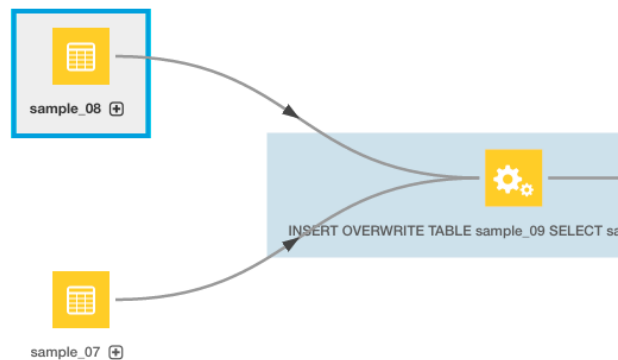
4. Optionally, click the **View Lineage** link in the entity details box to view the lineage of the selected entity.

Displaying a Template Lineage Diagram

A ***template lineage diagram*** contains template entities, such as jobs and queries, that can be instantiated, and the input and output entities to which they are related.

To display a template lineage diagram:

1. Perform a metadata search.

2. In the list of results, click an entity. The entity Details page displays. For example, when you click the `sample_09` result entry:

Hive sample_09
Type: **Table**   Parent Path: /default   Path: hdfs://tcdn1-1.ent.cloudera.com:8020/user/hive/warehouse/sample_09   Owner: hdfs
Created: Apr 8 2015 11:04 AM   Source: Hive

the Search screen is replaced with a Details page that displays the entity property sheet:
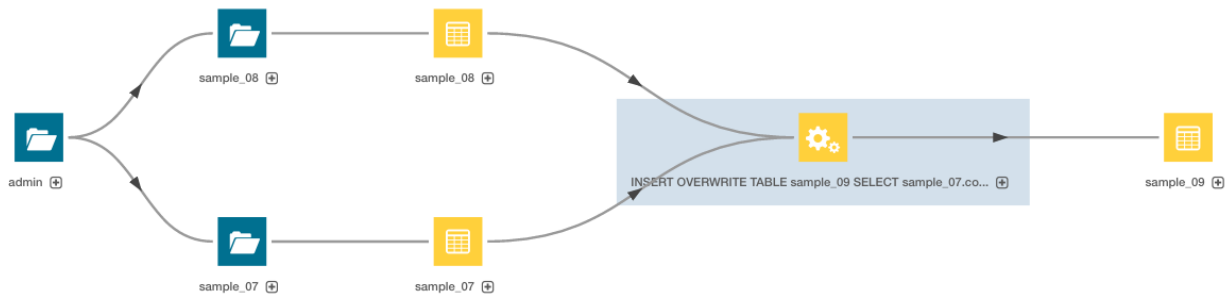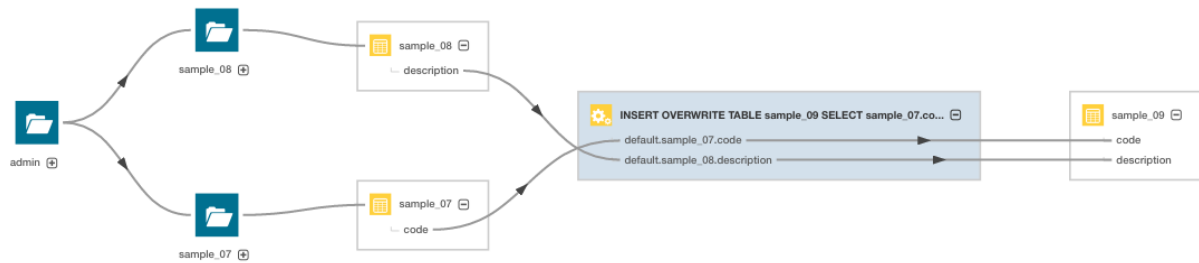

3. Click the **Lineage** tab. For example, clicking the Lineage tab for the sample_09 table displays the following lineage diagram:

This example shows the relations between a Hive query execution entity and its source and destination tables:

When you click the ⊕ icon, columns and lines connecting the source and destination columns display:

Displaying an Instance Lineage Diagram

An *instance lineage diagram* displays instance entities, such as job and query executions, and the input and output entities to which they are related. To display an instance lineage diagram:

1. Perform a search and click a link of type Operation.
2. Click a link in the **Instances** box.
3. Click the **Lineage** tab.

Displaying the Template Lineage Diagram for an Instance Lineage Diagram

To browse from an instance diagram to its template:

1. Display an instance lineage diagram.
2. Click the **Details** tab.
3. Click the value of the **Template** property to go to the instance's template.

Schema

Minimum Required Role: Lineage Administrator (also provided by **Metadata Administrator**, **Full Administrator**)

A table schema contains information about the names and types of the columns of a table.

A Kite dataset ingested into HDFS contains information about the names and types of the fields in an HDFS Avro or Parquet file used to create the dataset.

Displaying Hive, Impala, and Sqoop Table Schema

1. Perform a metadata search for entities of source type **Hive** and type **Table**.
2. In the list of results, click a result entry. The table schema displays in the Details tab.

Displaying Pig Table Schema

1. Perform a metadata [search](#) for entities of source type **Pig**.
2. In the list of results, click a result entry of type **Table**. The table schema displays in the Details tab.

Displaying HDFS Dataset Schema

If you ingest a [Kite dataset](#) into HDFS, you can view the schema of the dataset. The schema is represented as an entity of type Dataset and is implemented as an HDFS directory.

For Avro datasets, primitive types such as null, string, int, and so on, are not separate entities. For example, if you have a record type with a field A that's a record type and a field B that's a string, the subfields of A become entities themselves, but B has no children. Another example would be if you had a union of null, string, map, array, and record types; the union has 3 children - the map, array, and record subtypes.

To display an HDFS dataset schema:

1. Perform a metadata [search](#) for entities of type **Dataset**.
2. Click a result entry. The dataset schema displays in the Details tab.

Stocks Schema

1. Use the Stocks Avro schema file:

```
{
  "type" : "record",
  "name" : "Stocks",
  "namespace" : "com.example.stocks",
  "doc" : "Schema generated by Kite",
  "fields" : [ {
    "name" : "Symbol",
    "type" : [ "null", "string" ],
    "doc" : "Type inferred from 'AAIT'"
  }, {
    "name" : "Date",
    "type" : [ "null", "string" ],
    "doc" : "Type inferred from '28-Oct-2014'"
  }, {
    "name" : "Open",
    "type" : [ "null", "double" ],
    "doc" : "Type inferred from '33.1'"
  }, {
    "name" : "High",
    "type" : [ "null", "double" ],
    "doc" : "Type inferred from '33.13'"
  }, {
    "name" : "Low",
    "type" : [ "null", "double" ],
    "doc" : "Type inferred from '33.1'"
  }, {
    "name" : "Close",
    "type" : [ "null", "double" ],
    "doc" : "Type inferred from '33.13'"
  }, {
    "name" : "Volume",
    "type" : [ "null", "long" ],
    "doc" : "Type inferred from '400'"
  } ]
}
```

and the `kite-dataset` command to create a Stocks dataset:

```
kite-dataset create dataset:hdfs:/user/hdfs/Stocks -s Stocks.avsc
```

The following directory is created in HDFS:

2. In search results, the Stocks dataset appears as follows:

3. Click the **Stocks** link. The schema displays at the right of the Details tab.

Each subfield of the Stocks record is an entity of type Field.

4. Then use the `kite-dataset` command to import structured data:

```
kite-dataset csv-import ./Stocks.csv dataset:hdfs:/user/hdfs/Stocks --no-header
```

where `Stocks.csv` is:

```
AAPL,20150206,120.02,120.25,118.45,118.93,43372000
AAPL,20150205,120.02,120.23,119.25,119.94,42246200
GOOG,20150304,571.87,577.11,568.01,573.37,1713800
GOOG,20150303,570.45,575.39,566.52,573.64,1694300
GOOG,20150302,560.53,572.15,558.75,571.34,2118400
GOOG,20150209,528,532,526.02,527.83,1264300
GOOG,20150206,527.64,537.2,526.41,531,1744600
GOOG,20150205,523.79,528.5,522.09,527.58,1844700
FB,20150304,79.3,81.15,78.85,80.9,28014500
```

```
FB,20150303,79.61,79.7,78.52,79.6,18567300
FB,20150302,79,79.86,78.52,79.75,21604400
FB,20150227,80.68,81.23,78.62,78.97,30635700
FB,20150226,79.88,81.37,79.72,80.41,31111900
TWTR,20150211,46.27,47.78,46.11,47.5,24747000
TWTR,20150210,47.35,47.39,45.57,46.26,32287800
TWTR,20150209,46.73,47.69,46.5,47.32,36177900
TWTR,20150206,46.12,48.5,45.8,48.01,102669800
TWTR,20150205,42.04,42.47,40.91,41.26,61997300
MSFT,20150304,43.01,43.21,42.88,43.06,25705800
MSFT,20150303,43.56,43.83,43.09,43.28,31748600
MSFT,20150302,43.67,44.19,43.55,43.88,31924000
MSFT,20150227,44.13,44.2,43.66,43.85,33807700
MSFT,20150226,43.99,44.23,43.89,44.06,28957300
ORCL,20150304,43.2,43.66,42.82,43.61,14663900
ORCL,20150303,43.83,43.88,43.17,43.38,10058700
ORCL,20150302,43.81,44.04,43.48,44.03,11091000
ORCL,20150227,43.77,44.11,43.68,43.82,9549500
ORCL,20150226,43.8,44.15,43.71,43.89,8519300
ORCL,20150225,43.83,44.09,43.38,43.73,11785400
```