HITACHI
Inspire the Next

Using Pentaho Data Integration (PDI) with Hitachi Content Platform (HCP)

Change log:

Date	Version	Author	Changes
09/04/2019	1.0	Christopher Morehouse	
10/08/2019	1.1	Christopher Morehouse	added Deleting HCP Object Metadata section

Contents

This page intentionally left blank.

Overview

Pentaho Data Integration (PDI) can be used to move objects to and from Hitachi Content Platform (HCP). Starting in PDI 8.3, we have introduced native steps that give you the ability to query HCP objects as well as read and write object metadata.

This allows us to create extract, transform, and load (ETL) solutions for batch file processing directly in the HCP environment.

Our intended audiences are ETL developers, business intelligence (BI) developers, storage administrators, or anyone with a background in database development, BI, or IT storage who is interested in working with files stored in HCP using PDI.

The information in this document covers the following versions:

Software	Version(s)
Pentaho	8.3
Other Software	Hitachi Content Platform

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

Terms You Should Know

Here are some terms you should be familiar with:

- **HCP**: Hitachi Content Platform, an object-based storage repository designed for unstructured data. More information is available at PDI and Hitachi Content Platform (HCP).
- **VFS**: virtual filesystem
- **VFS browser**: virtual filesystem browser, a single browser for accessing various file systems. It presents a uniform view of files from various sources, such as the files on local disk in a Linux or Windows environment, on an HTTP server, inside a Zip archive, or an HCP environment. More information is available at Virtual file system browser.

Other Prerequisites

This document assumes that you have knowledge of Pentaho Data Integration and that you have already installed Spoon on a local machine. It also assumes you have access to an HCP environment and are familiar with basic HCP administration and use.

Use Cases

Use cases employed in this document include the following:

Use Case 1: Moving Objects to and from HCP

As part of a Pentaho ETL process, Company A needs to move CSV files to and from the Hitachi Content Platform. They will also need to delete files from HCP as dictated by their workflows.

Use Case 2: Querying and Extracting Data from Objects Stored in HCP

Company A has a process that automatically drops CSV files into the Hitachi Content Platform. They need to query for specific files, extract data from these files, and load the data into a relational database for reporting and analytics.

Interact with Objects Stored in HCP

To interact with objects that are stored in HCP, you will first need to configure your Pentaho environment. Then you will be able to move objects and extract data.

More information is available in these sections:

Configuring Pentaho Environment to Use HCP VFS URLs

You can use any PDI steps that take a configurable file path to interact with objects stored in HCP, if you are using an HCP VFS URL.

To get ready to use VFS URLs in PDI, create a `credentials` file that PDI will use to authenticate with HCP.

1. Create a `.hcp` folder under your user's operating system (OS) home directory and place the credentials file there.
    a. For Linux: `/home/user/.hcp/credentials`
    b. For Windows: `C:\Users\user\.hcp\credentials`
2. Place this content in the `credentials` file:

```
[default]
hcp_username=yourhcpusername
hcp_password=yourpassword
accept_self_signed_certificates=true
```

Moving Objects to and from HCP

Within PDI, at the job level, you can move objects to and from HCP.

Some of the job steps you can use for this include, but are not limited to:

- **Copy files**
- **Create a folder**
- **Delete files**
- **Move files**

Extracting Data from Objects Stored in HCP

Within PDI, at the transformation level, you can extract data directly from objects stored in HCP.

Some of the transformation steps you can use for this include, but are not limited to:

- **CSV file input**
- **Get file names**
- **Text file input**

Example PDI Step Configuration

As an example for PDI jobs, here is how to configure the **Copy files** step to copy CSV files from a local environment to a folder (in this case, `chris_test`) on HCP.

The **Destination File/Folder** is `hcp://nstest.r2d2.hcpdemo.pentaho.net/chris_test`.

HCP VFS URLs are constructed in the following format:
`hcp://namespace.tenant.hostname/foldername`.

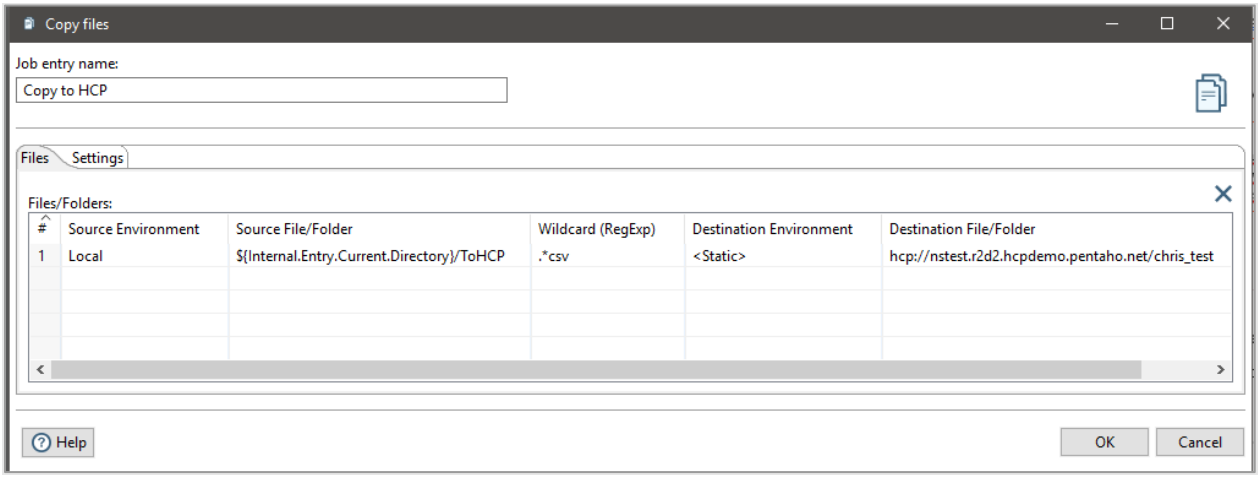

Figure 1: Copy Files Step Configuration

Query and Interact with HCP Object Metadata

Beginning with PDI 8.3, we have introduced a new feature called VFS connections. In addition, we have introduced the following transformation steps:

- **Query HCP**
- **Read metadata from HCP**
- **Write metadata to HCP**

More details are available in the following sections:

Configuring VFS Connections

Before you can use the new PDI steps you must first create a VFS connection.

Earlier, we discussed the use of VFS URLs being used in steps where file paths can be configured. **VFS connections** are different. They are a stored set of VFS properties that you can use to connect to a specific filesystem.

This is important to understand since the HCP steps that we will discuss use these preconfigured VFS connections rather than a manually constructed VFS URL.

You do not *need to set up the HCP VFS URLs `credentials` file if you are instead going to use VFS connections.*

Detailed instructions on how to configure VFS connections can be found at Set up a VFS connection in PDI.

Writing HCP Object Metadata

Any object stored in HCP can have custom metadata written to it. By default, only system metadata is created when an object is uploaded to HCP. We can write custom metadata annotations using the **Write metadata to HCP** step. Details about this step can be found at Write metadata to HCP.

In the following example, the first important thing to note is the HCP VFS connection that is selected. As discussed in the previous section, you will need to create a VFS connection first before using any of the HCP steps.

Here is what the example step will look like once it is configured. The procedure for filling out the properties follows.

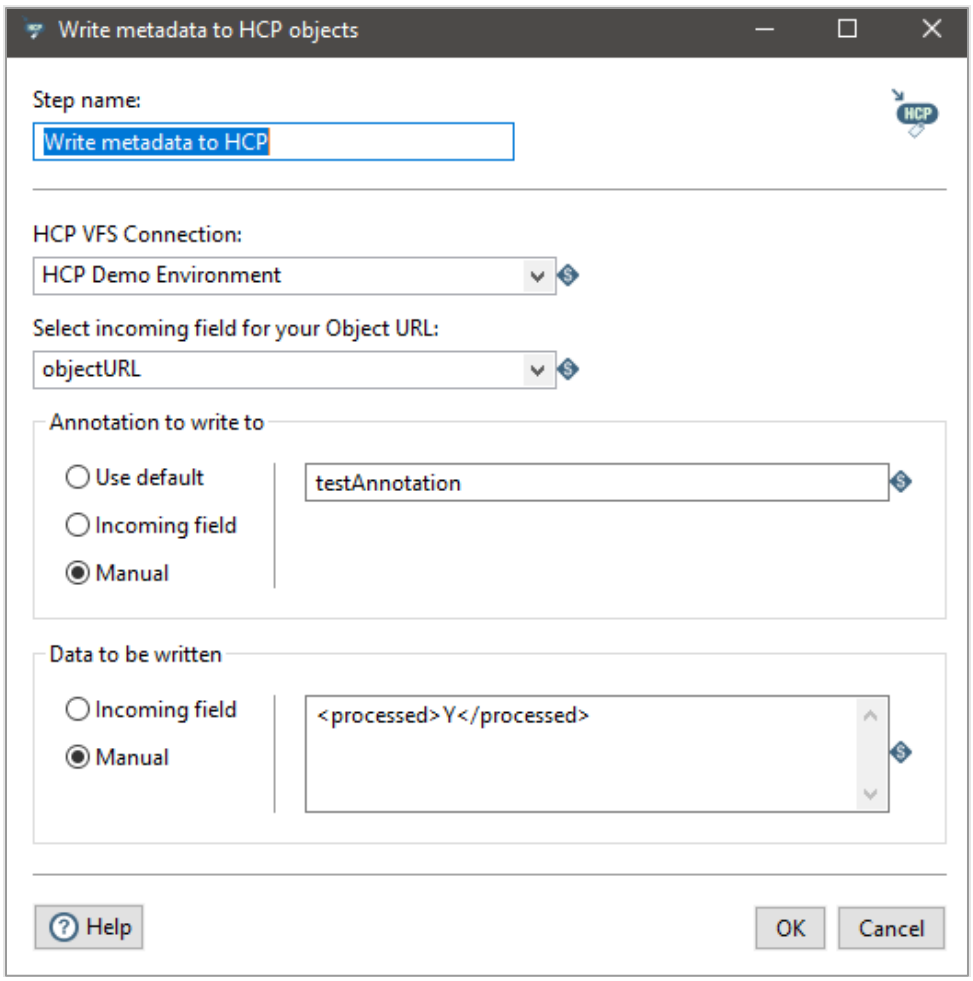

*Figure 2: **Write metadata to HCP** Step*

Continuing with this example:

1. Edit the **Write metadata to HCP** step.
2. Choose the **HCP VFS Connection** that you created (in this case, `HCP Demo Environment`).
3. In **Annotation to write to**, select **Manual** and name it `testAnnotation`. This will contain the custom metadata.
4. In **Data to be written**, select **Manual** and include your custom metadata as XML.

Metadata needs to be sent in XML format. If you do not use well-formed XML, metadata will not get written and an `Invalid XML` error will occur.

5. View the objects in the HCP web browser. Those that have custom metadata annotations will

have a clickable metadata icon as shown. Click on it.

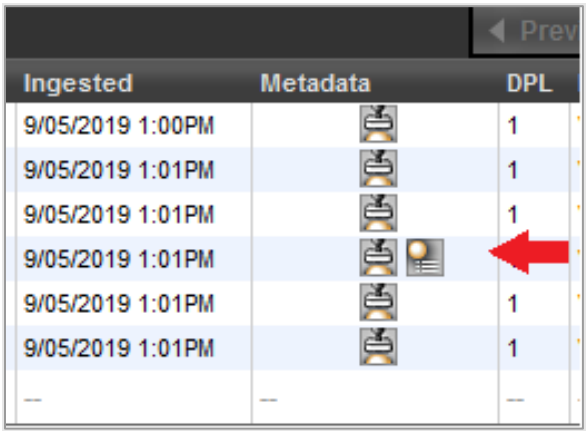

Figure 3: Custom Metadata Annotation Icon

6. In this example, when you click on the metadata icon, you can see the annotation that was written by the **Write metadata to HCP** step:

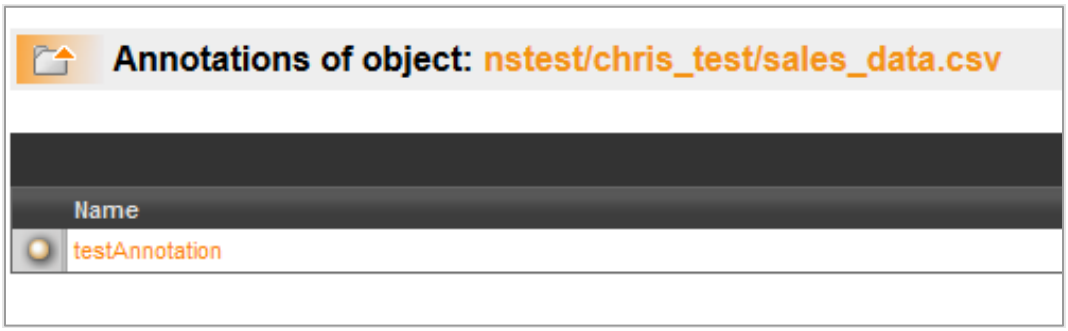

Figure 4: `testAnnotation` on Object

7. Click on the annotation to reveal the actual metadata:

Figure 5: XML Metadata

Reading HCP Object Metadata

We can read custom HCP object metadata by using the **Read metadata from HCP** step. Details about this step can be found in our official documentation in Read metadata from HCP.

In this example, we will read the `testAnnotation` metadata that was written in the previous section:

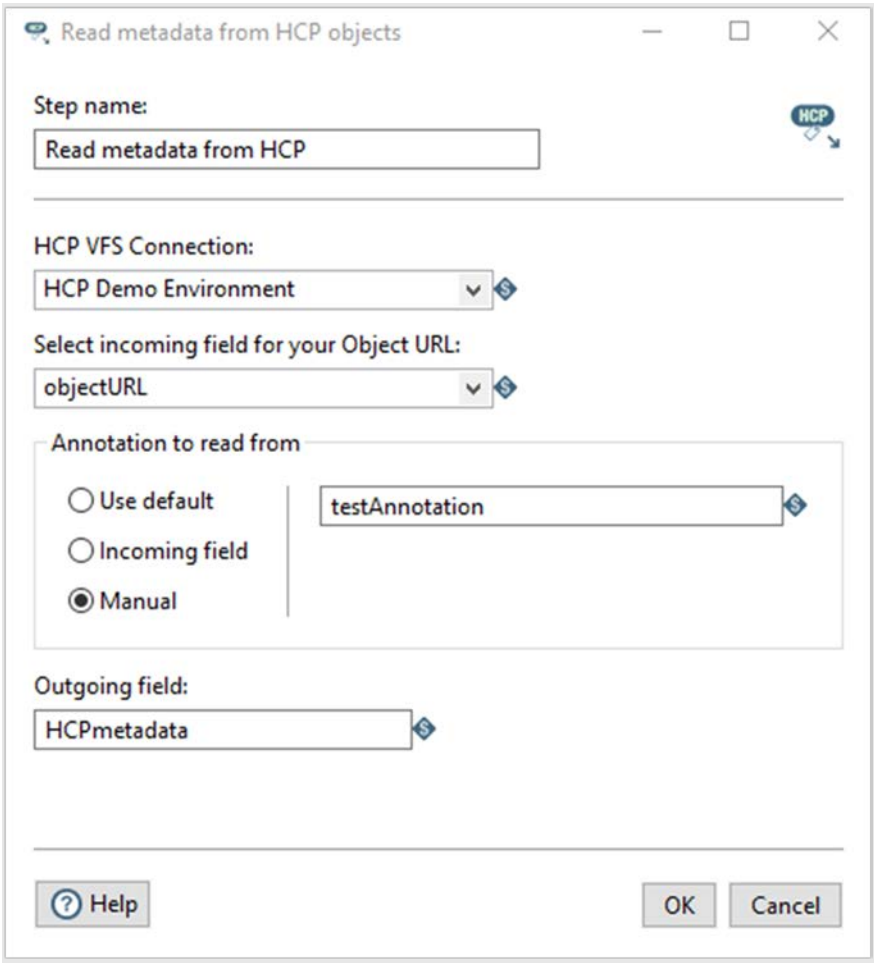

*Figure 6: Configuring **Read metadata from HCP** Step*

To set up and use this step:

1. Edit a **Read metadata from HCP** step and configure it as shown.
2. Preview the step to show the custom metadata that was written previously.
3. Notice how the **objectURL** is an `https://` URL instead of the `hcp://` VFS URL:

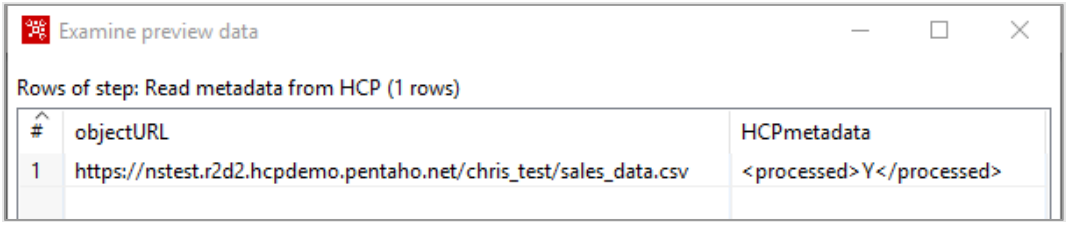

Figure 7: Examine Preview Data

Querying HCP Objects

We can query HCP objects using the **Query HCP** step and the HCP Metadata Query Engine. Details about this step can be found in our official documentation on Query HCP.

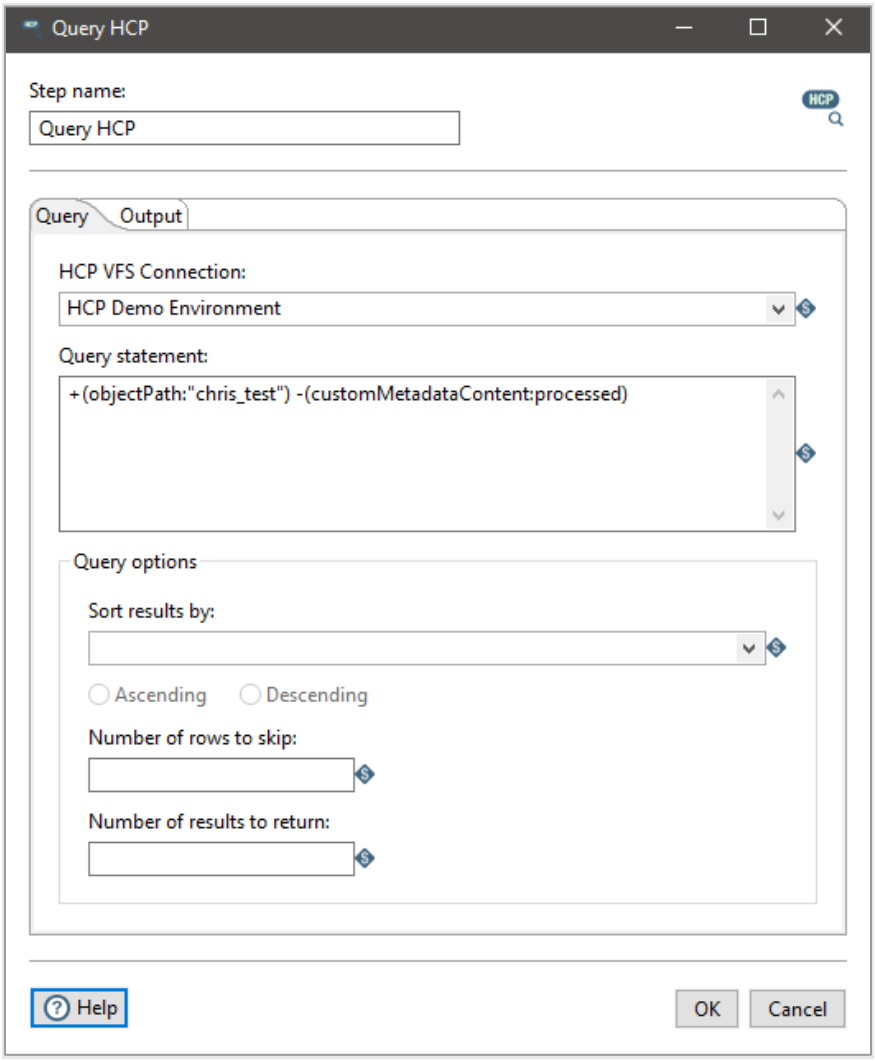

Figure 8: Query HCP Setup Example

As you can see on the **Query** tab, this step has been configured using the following Metadata Query statement:

```
+(objectPath:"chris_test") -(customMetadataContent:processed)
```

This statement is saying, "Get objects located within a folder named `chris_test`, and only objects that do not have a `processed` metadata annotation." See the next section, How to Construct an HCP Metadata Query Statement, for instructions on how to build a query statement.

Under the **Output** tab of the step, the **Outgoing field for Object URL** is configured with the HCP Object URL.

Return all object properties is also checked. These object properties are the default system metadata that gets written to an object when it is first uploaded to HCP.

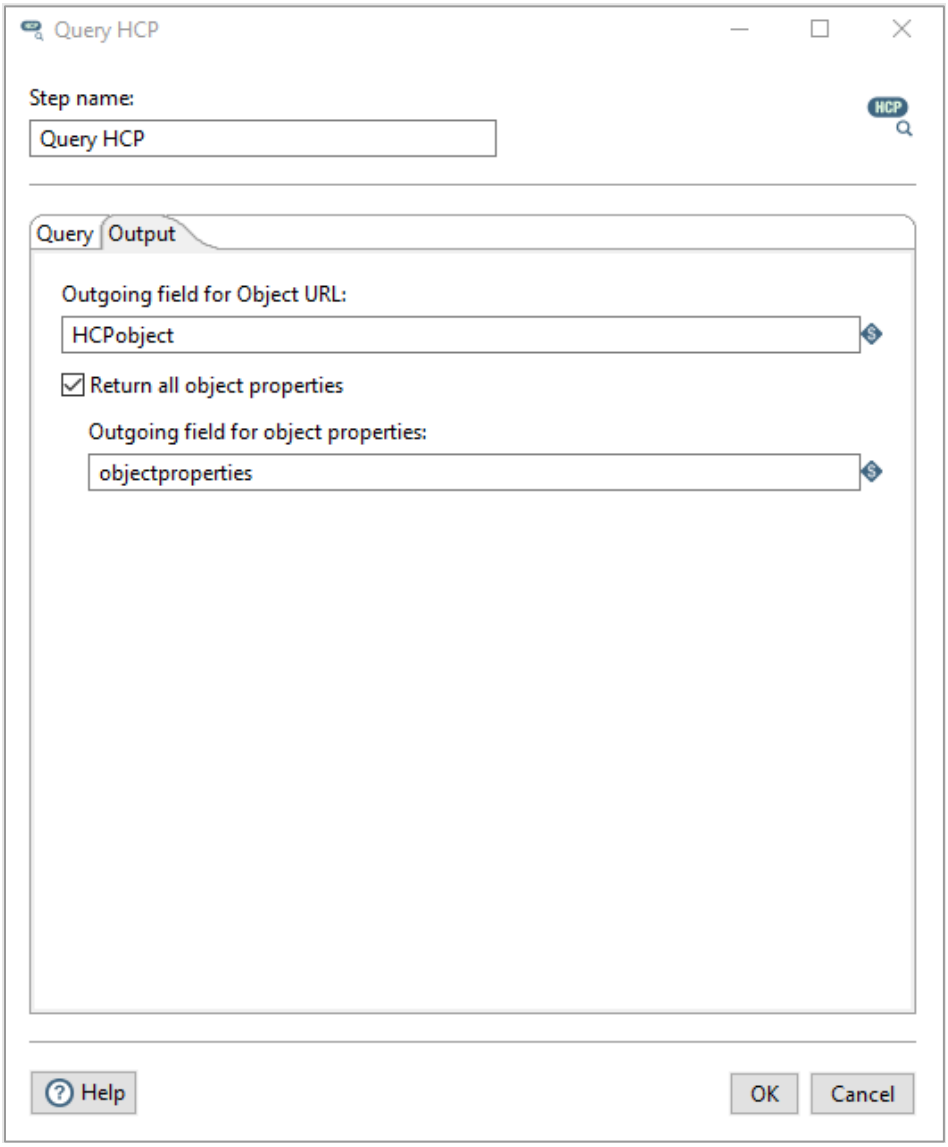

Figure 9: Query HCP – Output Tab

Previewing this step shows only one object that does not have the `processed` metadata set:

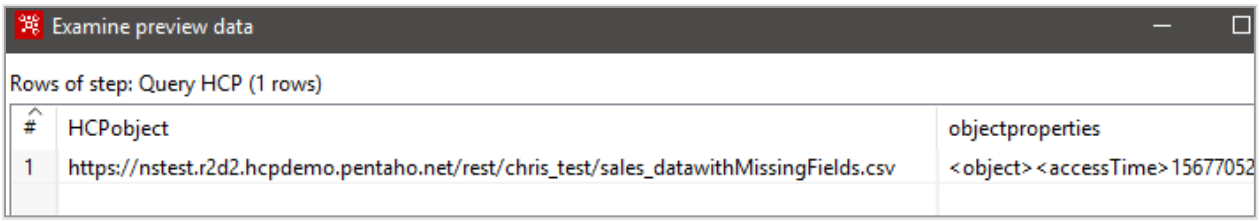

Figure 10: Preview Data of Query HCP

How to Construct an HCP Metadata Query Statement

You do not need to know the specifics of how to construct a metadata query to use the **Query HCP** step.

A simple way to create query statements is to use the query builder located in the HCP web browser:

1. In the HCP Browser, click on the **Search** icon in the upper right-hand corner.
2. Click on the **Structured Query** tab.
3. Construct your query by adding as many conditions as necessary.
4. Click on **Show as advanced query** towards the bottom.
5. Copy the resulting query statement to the **Query HCP** PDI step.

As an example, here is the previous query statement built using the Structured Query module:

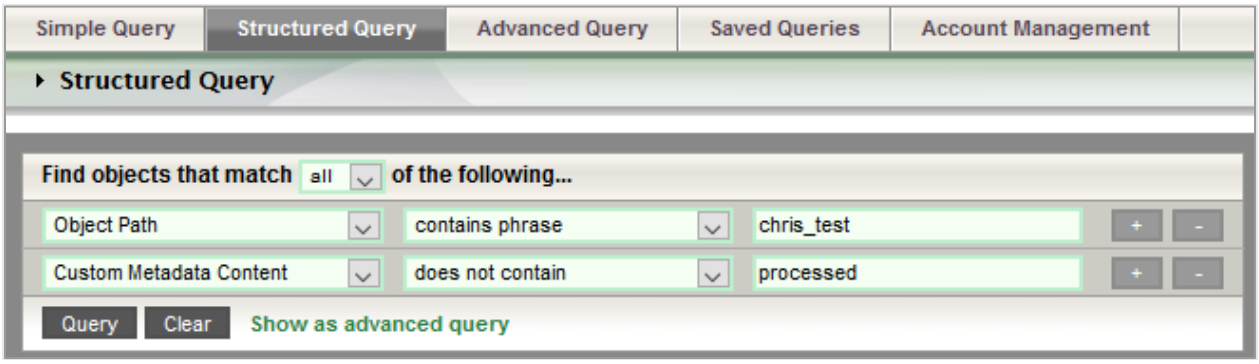

Figure 11: Query Builder in HCP Web Browser

When you click on **Show as advanced query** and then submit the query, you see the following:

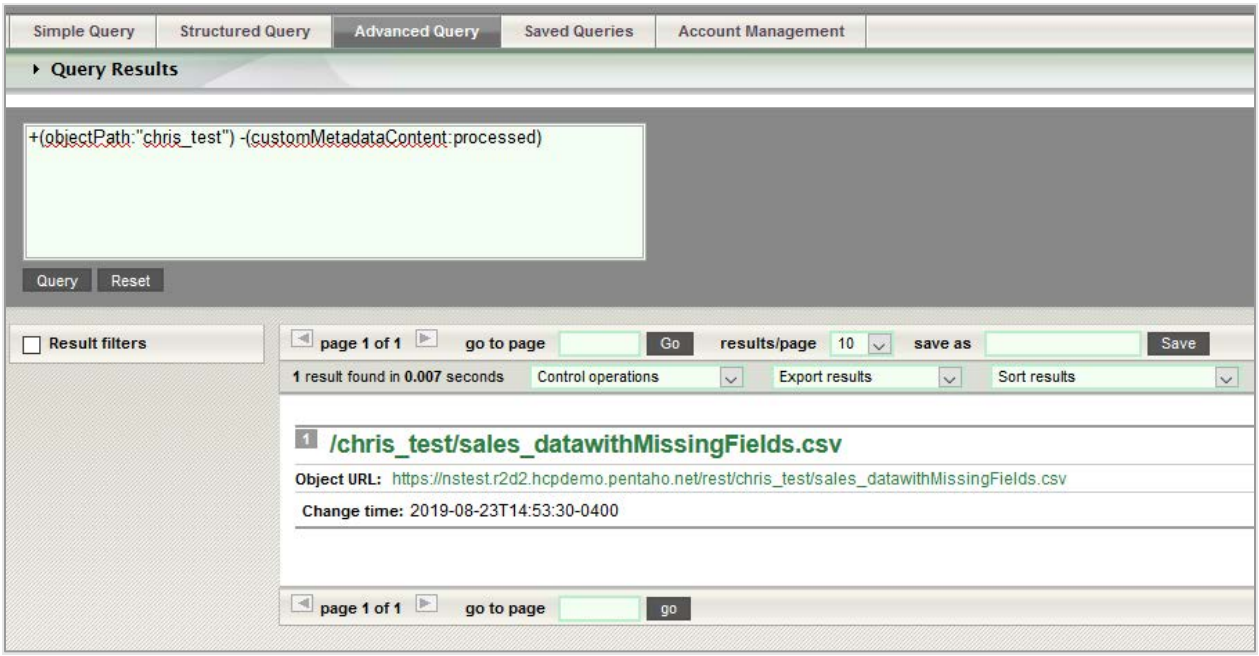

Figure 12: Query Results

Deleting HCP Object Metadata

Unlike the previous options we explored here, there is no native HCP step that will delete HCP metadata annotations.

However, you can use HCP's REST API and the PDI **REST client** transformation step to complete this task instead. More information on the **REST client** step can be found here in our documentation at REST Client.

In this example, we will use two fields with the following values:

```
authenticationString = HCP Y21vcmVob3VzZQ==:5f4dcc3b5aa765d61d8327deb882cf99
```

and

```
url =
https://nstest.r2d2.hcpdemo.pentaho.net/rest/chris_test/sales_data.csv?type=custom
-metadata&annotation=testAnnotation
```

To delete metadata annotations:

1. Note the authentication token used in `authenticationString`. Use the following format to create your authentication token:
   `HCP <base64-encoded-username>:<md5-hashed-password>`
2. Use the following format for your URL:
   `https://namespace.tenant.hostname/rest/foldername/objectname?type=custo
   m-metadata&annotation=annotationname`

3. Configure the **REST client** step as shown:

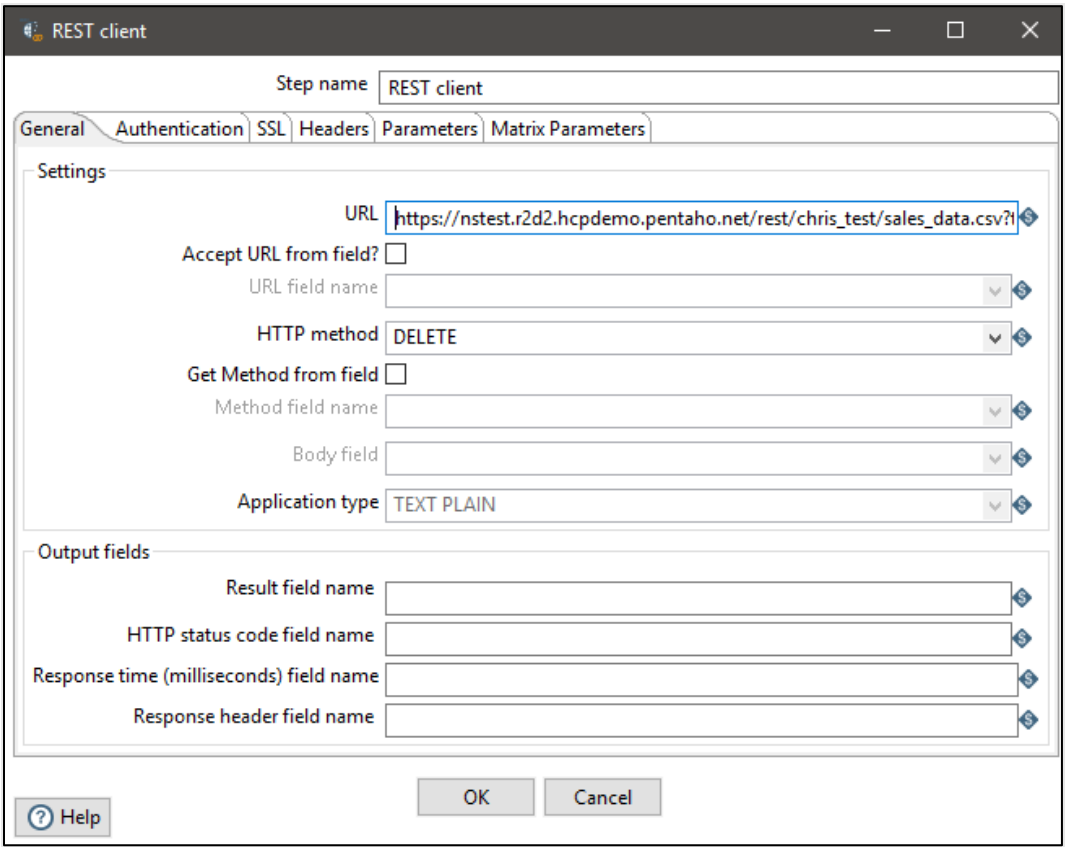

Figure 13: REST client

4. Pass the `authenticationString` to the **Authorization** header under the **Headers** tab:

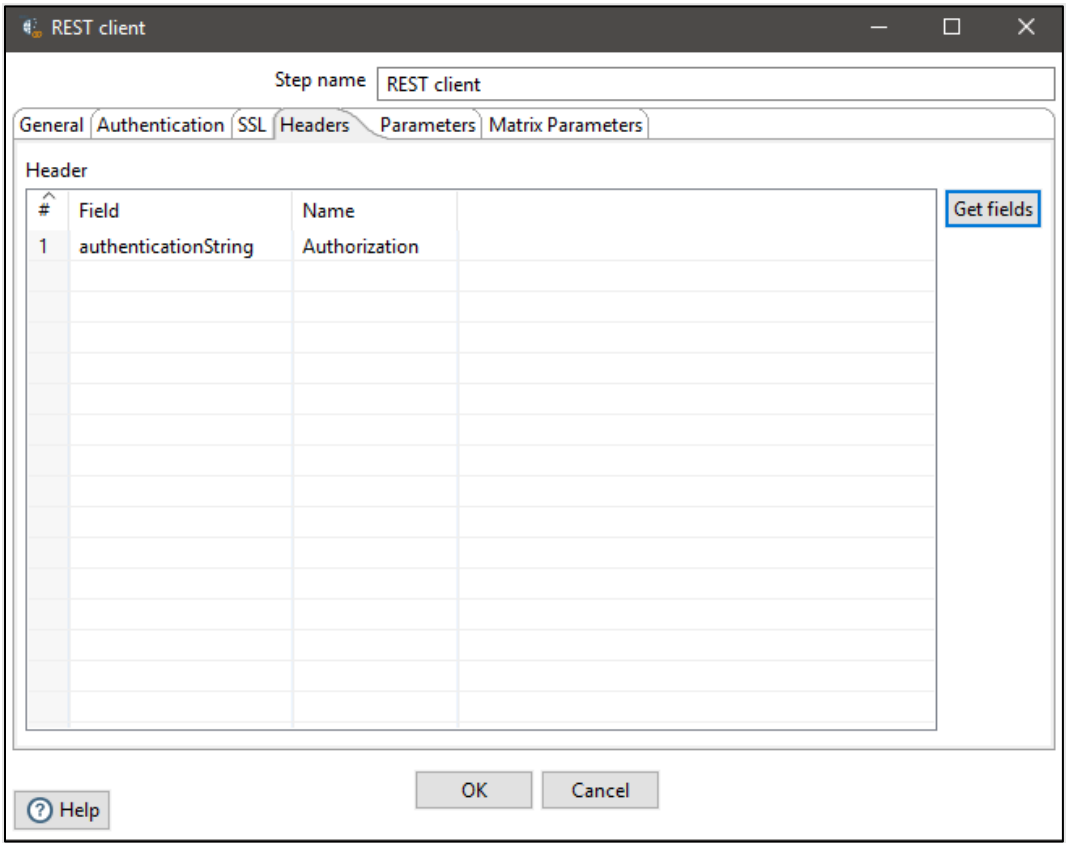

Figure 14: Headers Tab

You may need to import your HCP instance's certificate to the Java keystore of your PDI client, as well as to your Pentaho Server, if you are using an HTTPS URL. Otherwise, you may get `javax.net.ssl.SSLHandshakeException` errors.

Known Behavior

There is some behavior that is important to know when developing your ETL with HCP. Information on this behavior and some workarounds are available in the following sections:

- Metadata Query Engine Delay
- Password Bug in VFS Connections

Metadata Query Engine Delay

When you first upload an object to HCP, there is a delay with when the HCP Metadata Query Engine will see it. That is, if you issue a query statement with either the **Query HCP** PDI step or the query builder in HCP's web browser, it can take up to two minutes for the object to appear in the query's result. This happens because it takes some time for the query engine to index the new object(s).

This delay also affects changes to the metadata. So, if you have an existing object in HCP and update or add new metadata, these changes will not appear in query results until they are indexed by the query engine.

Workaround

Since this delay only affects queries, but new objects and updated metadata are available immediately using HCP's REST API as well as the HCP web browser, we could use the **Read metadata from HCP** step as shown:

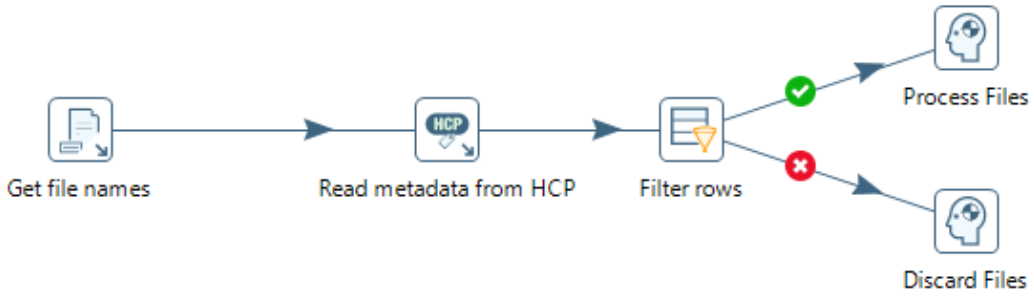

Figure 15: Read metadata from HCP

In this example:

1. Use the **Get file names** step to get a list of HCP file names to process (using an HCP VFS URL).
2. Then, use the **Read metadata from HCP** step to see if there is `processed` metadata configured.
3. Use the **Filter rows** step to ignore files that have `processed` metadata and process any files that have a null annotation or ones where the metadata has indicated the file has not been processed.

Obviously, this example would only work in a scenario where the ETL is adding `processed` metadata at the end of an execution.

*This workaround is not recommended for large amounts of files since the **Read metadata from HCP** step performs slower than the **Query HCP** step. Careful consideration is required before using this type of architecture.*

Password Bug in VFS Connections

At the time of this writing, there is a bug in PDI 8.3.0.0 with the VFS connections screen in PDI where passwords are in plaintext.

Workaround

A Service Pack will be released to fix this bug, so upgrading to the latest version of Pentaho will address this. At the time of this writing, this bug only affects version 8.3.0.0. Please contact Support for the exact version where this bug has been addressed.

Alternatively, you can use a variable in the **Password** field and include the password in `kettle.properties`. Here is an example where we are using the variable `${hcp_password}`:

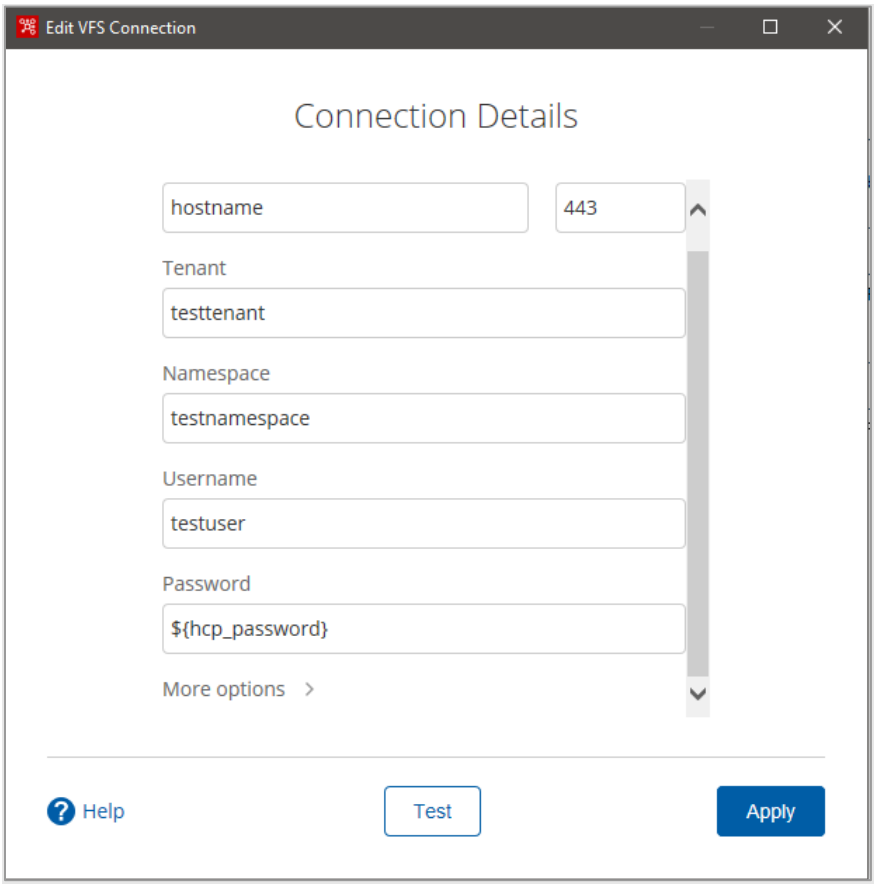

Figure 16: Edit VFS Connection

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- Apache Commons VFS Documentation
- Components Reference
- PDI and Hitachi Content Platform (HCP)
- Pentaho Virtual File System Browser
- Query HCP
- Read Metadata from HCP
- REST Client
- Set Up a VFS Connection in PDI
- Write Metadata to HCP