Informatica

Informatica® PowerExchange for HBase 10.5.6

User Guide

Informatica PowerExchange for HBase User Guide
10.5.6
May 2024

Publication Date: 2024-05-30

Table of Contents

Preface

Use the *Informatica® PowerExchange® for HBase User Guide* to learn how to read from or write to column families in an HBase table by using the Developer tool. Learn to create a connection and develop and run mappings and in the native environment and Hadoop environments.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction to PowerExchange for HBase

This chapter includes the following topic:

PowerExchange for HBase Overview

PowerExchange for HBase provides connectivity to an HBase data store.

Use PowerExchange for HBase to read data from the HBase columns families or write data to the columns families in an HBase table. You can read or write data to a column family or a single binary column.

You can connect to an HBase data store and view all the HBase tables. Select an HBase table and view all the column families. If you know the schema of the data source, you can specify the columns in the column families when you create the data object. If you do not know the schema of the data source, you can search the rows in an HBase table to identify the columns and their occurrence probability.

You can validate and run mappings in the native environment or a Hadoop environment.

Example

You work for a mobile service provider that needs to load the data in WAP log files to HBase tables and generate multiple reports. WAP log files can contain a number of columns with information about the mobile users, internet usage, and data volume. On any day, the WAP log files could be three to four billion rows of data and can be around two terabytes in size.

You can use PowerExchange for HBase to consolidate data received during the day, filter and transform the data, and load it to HBase tables. Based on the data in the tables, analysts can run queries and perform real-time analysis of daily operations, statistics related of gateway usage, and data volume.

CHAPTER 2

PowerExchange for HBase Configuration

This chapter includes the following topic:

Prerequisites

Before you use PowerExchange for HBase, perform the following tasks:

- Install and configure Informatica Services. Verify that the domain has a Data Integration Service and a Model Repository Service.
- Verify that a cluster configuration is created in the domain.
- Verify that if you are the machine login user in case of non-Kerberos or the keytab user in case of Kerberos in Ranger, you should have the create and read privileges on the HBase table that you want to import.
- Verify that a Metadata Access Service is created in the domain.
- Verify that the Hadoop Distribution Directory property in the developerCore.ini file is set based on the Hadoop distribution that you use.

CHAPTER 3

HBase Connections

This chapter includes the following topics:

HBase Connections Overview

Create an HBase connection to read data from or write data to an HBase table.

Use the Developer tool, Administrator tool, or Analyst tool to create the connections.

HBase Connection Properties

Use an HBase connection to access HBase. The HBase connection is a NoSQL connection. You can create and manage an HBase connection in the Administrator tool or the Developer tool. HBase connection properties are case sensitive unless otherwise noted.

The following table describes HBase connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: ~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.

Property	Description
Description	The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select HBase.
Database Type	Type of database that you want to connect to. Select **HBase** to create a connection for an HBase table.

Creating an HBase Connection

Create an HBase connection before you create an HBase data object.

1. In the Developer tool, click **Window** > **Preferences**.
2. Select **Informatica** > **Connections**.
3. Expand the domain in the **Available Connections** list.
4. Select the connection type **NoSQL** > **HBase**, and then click **Add**.
5. Enter a connection name.
6. Optionally, enter a connection ID and description.
7. Select the domain where you want to create the connection.
8. Select **HBase** as the connection type.
9. Click **Next**.
10. Select the database type as **HBase**.
11. Select the cluster configuration associated with the Hadoop environment.
12. Click **Test Connection**. If a default Metadata Access Service is not set, a message appears to configure the Metadata Access Service. Click **OK** and set one Metadata Access Service as default. After you set a default Metadata Access Service, the connection to the HBase table is tested. If the Metadata Access Service does not exist, contact the Informatica administrator to create a new Metadata Access Service.
13. Click **Finish**.

Troubleshooting an HBase Connection Configured for Clusters that Do Not Use Kerberos Authentication

I got a valid Kerberos ticket when I configured a Kerberos connection in the Developer client to connect to the HBase master server that uses Kerberos authentication. Later when I used the same setup and disabled the Kerberos authentication to connect to a cluster that does not use secure authentication, the connection failed.

The error occurs because the mode of authentication in the `hive-site.xml` configuration set does not update and the same ticket that was generated for a Kerberos cluster is used for the cluster that does

not use Kerberos authentication. To resolve the issue, verify that the `hive-site.xml` configuration set contains the correct mode of authentication.

CHAPTER 4

HBase Data Objects

This chapter includes the following topics:

HBase Data Objects Overview

An HBase data object is a physical data object that represents data based on an HBase resource. After you create an HBase connection, create an HBase data object and a data object operation.

When you create an HBase Data Object, you can select an HBase table and view all the column families in the table. You can specify the column names in the column family if you know the column name and data type, or you can search the rows in the HBase table and specify the columns.

You can read from and write to a column family or to a single binary column. When you create the data object, specify the column families to read or choose to get all the data as a single stream of binary data. You can also specify the column families to which you can write or choose to write all the data as a single stream of binary data.

Create a data object read operation to read data from the HBase column families and create a data object write operation to insert data to the HBase column families.

Data Object Column Configuration

When you want to read data from or write data to columns in a column family, you can specify the columns when you create the HBase data object.

You can add the columns in the column families or you can search for the columns names in the column family and select the columns. You can also choose to read from or write to a single binary port.

Add Columns

When you create a data object, you can specify the columns in one or more column families in an HBase table.

When you add an HBase table as the resource for an HBase data object, all the column families in the HBase table appear. If you know the details of the columns in the column families, you can select a column family and add the column details. Column details include column name, data type, precision, and scale.

Although data is stored in binary format in HBase tables, you can specify the associated data type of the column to transform the data. To avoid data errors or incorrect data, verify that you specify the correct data type for the columns.

Verify that you specify valid column details when you add columns to avoid unexpected run-time behaviors. If you add a column that does not exist in the column family and create a data object read operation, the Data Integration Service returns a null value for the column at run time. If you do not specify a value for a column when you write data to an HBase table, the Data Integration Service specifies a null value for the column at run time.

If the HBase table has more that one column family, you can add column details for multiple column families when you create the data object. Select one column family at a time and add the columns details. The column family name is the prefix for all the columns in the column family for unique identification.

Search and Add Columns

When you create a data object, you can search the rows in an HBase table to identify the column in the table and select the columns you want to add.

When you do not know the columns in an HBase table, you can search the rows in the table to identify all the columns and the occurrence percentage of the column. You can infer if the column name is valid based on the number of times the column occurs in the table. For example, if column name eName occurs rarely while column name empName occurs in a majority of rows, you can infer the column name as empName.

When you search and add columns, you can specify the maximum number of rows to search and the occurrence percentage value for a column. If you specify the maximum numbers of rows as 100 and the column occurrence percent as 90, all columns that appear at least 90 times in 100 rows appear in the results. You can select the columns in the results to add the columns to the data object.

Get All Columns

Binary data or data that can be converted to a byte array can be stored in an HBase column. You can read from and write to an HBase tables in bytes.

When you create a data object, you can choose to get all the columns in a column family as a single stream of binary data.

Use the HBase data object as a source to read data in all the columns in the column family as a single stream of binary data. Use the HBase data object as a target to write data in all the columns in the source data object as a single column of binary data in the target HBase table.

The Data Integration Service generates the data in the binary column based on the protobuf format. Protobuf format is an open source format to describe the data structure of binary data. The protobuf schema is described as messages.

HBase Data Object Properties

Specify the data object properties when you create the data object.

General Properties

The following table describes the general properties that you configure for the HBase data objects:

Property	Description
Name	Name of the HBase data object.
Location	The project or folder in the Model repository where you want to store the HBase data object.
Connection	Name of the HBase connection.

Add Column Properties

In the **Column Families** dialog box, select the column family to which you want to add the columns. The following table describes the column properties that you configure when you associate columns with column families:

Property	Description
Name	Name of the column in the column family.
Type	Data type of the column.
Precision	Precision of the data.
Scale	Scale of the data.

Search and Add Column Properties

The following table describes the column properties that you configure when you search for columns in column families and add the required columns:

Property	Description
Maximum rows to sample	Maximum number of rows in the HBase table you want to include while searching for columns. Default is 100.
Column occurrence percent	The threshold occurrence percentage of the column. A column appears in the results when the occurrence percentage value of the column meets or exceeds the threshold value. Default is 90.

HBase Data Object Read Operation Properties

HBase data object read operation properties include run-time properties that apply to the HBase data object.

The Developer tool displays advanced properties for the HBase data object operation in the Advanced view.

The following table describes the Advanced properties for an HBase data object read operation:

Property	Description
Tracing Level	Amount of detail that appears in the log for this transformation. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Maintain row order	Maintain the row order of the input data to the transformation. Select this option if the Data Integration Service should not perform any optimization that can change the row order. When the Data Integration Service performs optimizations, it might lose an order established earlier in the mapping. You can establish order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a transformation to maintain row order, the Data Integration Service considers this configuration when it performs optimizations for the mapping. The Data Integration Service performs optimizations for the transformation if it can maintain the order. The Data Integration Service does not perform optimizations for the transformation if the optimization would change the row order.
Date Time Format	Format of the columns of the date data type. Specify the date and time formats by using any of the Java date and time pattern strings.
Auto Flush	Optional. Indicates whether you want to enable Auto Flush. You can set auto flush to the following values: - Enable **Auto Flush** to set the value to true. The Data Integration Service runs each Put operation immediately as it receives them. The service does not buffer or delay the Put operations. Operations are not retried on failure. When you enable auto flush, the operations are slow as you cannot run operations in bulk. However, you do not lose data as the Data Integration Service writes the data immediately. - Disable **Auto Flush** to set the auto flush value to false. When you disable auto flush, the Data Integration Service accepts multiple Put operations before making a remote procedure call to perform the write operations. If the Data integration Service stops working before it flushes any pending data writes to HBase, that data is lost. Disable auto flush if you need to optimize performance. Default is disabled.
Default Column Data Type	Data type of the additional ports generated at run time.
Default Precision	Precision of the additional ports generated at run time.
Default Scale	Scale of the additional ports generated at run time.
Default Column Family	Column family name generated at run time. Required if the HBase target columns are defined based on the mapping flow.
Control File Location	Path and file name of the control file. Required if you generate run-time column names from the control file.

HBase Data Object Write Operation Properties

HBase data object write operation properties include run-time properties that apply to the HBase data object.

The Developer tool displays advanced properties for the HBase data object operation in the Advanced view.

The following table describes the Advanced properties for an HBase data object write operation:

Property	Description
Tracing Level	Amount of detail that appears in the log for this transformation. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Maintain row order	Maintain the row order of the input data to the transformation. Select this option if the Data Integration Service should not perform any optimization that can change the row order. When the Data Integration Service performs optimizations, it might lose an order established earlier in the mapping. You can establish order in a mapping with a sorted flat file source, a sorted relational source, or a Sorter transformation. When you configure a transformation to maintain row order, the Data Integration Service considers this configuration when it performs optimizations for the mapping. The Data Integration Service performs optimizations for the transformation if it can maintain the order. The Data Integration Service does not perform optimizations for the transformation if the optimization would change the row order.
Date Time Format	Format of the columns of the date data type. Specify the date and time formats by using any of the Java date and time pattern strings.
Auto Flush	Optional. Indicates whether you want to enable Auto Flush. You can set auto flush to the following values: - Enable **Auto Flush** to set the value to true. The Data Integration Service runs each Put operation immediately as it receives them. The service does not buffer or delay the Put operations. Operations are not retried on failure. When you enable auto flush, the operations are slow as you cannot run operations in bulk. However, you do not lose data as the Data Integration Service writes the data immediately. - Disable **Auto Flush** to set the auto flush value to false. When you disable auto flush, the Data Integration Service accepts multiple Put operations before making a remote procedure call to perform the write operations. If the Data integration Service stops working before it flushes any pending data writes to HBase, that data is lost. Disable auto flush if you need to optimize performance. Default is disabled.
Default Column Data Type	Data type of the additional ports generated at run time.
Default Precision	Precision of the additional ports generated at run time.
Default Scale	Scale of the additional ports generated at run time.
Default Column Family	Column family name generated at run time. Required if the HBase target columns are defined based on the mapping flow.
Control File Location	Path and file name of the control file. Required if you generate run-time column names from the control file.

Parameterization of HBase Data Objects

You can parameterize the HBase connection and the HBase data object operation properties.

You can parameterize the following data object read operation properties for HBase data objects:

- Connection in the run-time properties
- Filter condition in the query properties
- Date Time Format in the advanced properties

You can parameterize the following data object write operation properties for HBase data objects:

- Connection in the run-time properties.
- Date Time Format in the advanced properties.

Creating an HBase Data Object

Create an HBase data object to specify an HBase resource.

1. Select a project or folder in the Object Explorer view.
2. Click **File** > **New** > **Data Object**.
3. Select **HBase Data Object** and click **Next**.

   The **New HBase Data Object** dialog box appears.
4. Enter a name for the data object.
5. Click **Browse** next to the **Location** option and select the target project or folder.
6. Click **Browse** next to the **Connection** option and select a connection from which you want to import the HBase resource.
7. From the **Available OS Profiles** list, select an operating system profile. You can use the **Available OS Profiles** to increase security and to isolate the design-time user environment when you import and preview metadata from a Hadoop cluster.

   Note: The Developer tool displays the **Available OS Profiles** list only if the Metadata Access Service is enabled to use operating system profiles. The Metadata Access Service imports the metadata with the default operating system profile assigned to the user. You can change the operating system profile from the list of available operating system profiles.
8. To add a resource to the data object, click **Add** next to the **Resource** option.

   If a default Metadata Access Service is not set, a message appears to configure the Metadata Access Service. Click **OK** and set one Metadata Access Service as default. After you set a default Metadata Access Service, the **Add Resource** dialog box appears. If the Metadata Access Service does not exist, contact the Informatica administrator to create a new Metadata Access Service.
9. Navigate or search for the resources to add to the data object and click **OK**.

   You can add one HBase table to the data object.
10. Click **Next**. The **Column Families** dialog box appears.
11. Select a column family and specify the columns in it. Choose to add columns or get all columns.
    - To manually add, or search and add columns to the column family, select the **Add Columns** option.

- To read from or write all columns in the column family to a single binary column, select the **Get all columns** option.

12. Add the columns in the column family. Choose to add columns or search the column names in the column family and add the columns.
    - To specify the columns from the column family when you know the column name and data type, select the column family to which you want to add the columns and click **Add**. Configure the add properties.
    - To search columns in the column family and add them, click **Search and Add**. The **Search and Add** dialog box appears.
13. Specify the following details in the **Search and Add** dialog box:
    a. Specify the maximum rows in the HBase tables you want to include in the search.
    b. Specify the threshold value of the column occurrence percentage.
    c. Click **Go**.
       The column name and the occurrence percentage of the column in the table appears in the results.
    d. Select the columns that you want to specify for the column family. Configure the add properties.
14. Click **Next.**
    The **Create Row** dialog box appears.
15. Select the **Include Row ID** option to generate a row ID for the HBase table.
16. Specify the data type, precision, and scale for the row ID and click **Next.**
    The **Review Columns** dialog box appears. The column family name is the prefix for all the column names in that column family for unique identification. Default data type of the row ID is String.
17. Review the columns in the column families and click **Finish**.
    The data object appears under Data Object in the project or folder in the Object Explorer view. You can also add resources to a data object after you create it.

Creating an HBase Data Object Operation

Create a data object operation from a data object.

Before you create a data object operation, you must create the data object with the resource.

1. Select the data object in the Object Explorer view.
2. Right-click and select **New** > **Data Object Operation**.
   The **Data Object Operation** dialog box appears.
3. Enter a name for the data object operation.
4. Select the type of data object operation. You can choose to create a read operation or a write operation.
5. Click **Add**.
   The **Select a resource** dialog box appears.
6. Select the resource for which you want to create the data object operation and click **OK**.
7. Click **Finish**.
   The Developer tool creates the data object operation for the selected data object.

CHAPTER 5

HBase Mappings

This chapter includes the following topics:

HBase Mappings Overview

After you create an HBase data object operation, you can create a mapping.

To read or lookup data from HBase, use a data object read operation based on the HBase resource. Configure the read operation properties and add the read operation as a Read transformation or Lookup transformation to a mapping.

To write data to HBase, use a data object write operation based on the HBase resource. Configure the write operation properties and add the write operation as a Write transformation to a mapping.

When you configure the data object columns, you can get data in all columns in a column family to a single column as binary data. Use the Data Processor transformation to convert the binary data into the required data types.

You can validate and run mappings in the native environment or in the Hadoop environment. When you run a mapping in the Hadoop environment, you can select the Blaze or Spark engine. The Data Integration Service then creates multiple Map jobs to read data or write data in parallel.

You can use HBase objects as dynamic sources and targets in a mapping. For information about dynamic mappings, see the *Informatica Developer Mapping Guide*.

HBase Dynamic Mapping Overview

You can use HBase data objects as dynamic sources and targets in a mapping.

Use the HBase dynamic mapping to accommodate changes to source, target, and transformation logics at run time. You can use an HBase dynamic mapping using control files to manage frequent schema or metadata changes or to reuse the mapping logic for data sources with different schemas.

If the data source for a source and target changes, you can configure a mapping to dynamically get metadata changes at runtime. If a source changes, you can configure the Read transformation to accommodate changes. If a target changes, you can configure the Write transformation accommodate target changes.

You do not need to manually synchronize the data object and update each write transformation to accommodate changes, before you run the mapping again. The Data Integration Service dynamically determines the transformation ports, transformation logic in the ports, and the port links within the mapping.

There are two options available to enable a mapping to run dynamically.

You can select one of the following options to enable the dynamic mapping:

- In the **Data Object** tab of the data object read or write operation, select the **At runtime, get data object columns from data source** option when you create a mapping.
  When you enable the dynamic mapping using this option, you can refresh the source and target schemas at the runtime.
- In the **Ports** tab of the data object write operation, select the value of the **Columns defined by** property as **Mapping Flow** when you configure the data object write operation properties.
  When you enable the dynamic mapping using this option, you can add all the Source transformation or transformation ports to the target dynamically and the Data Integration Service creates a target file with the ports at runtime.

Note: Dynamic mapping is applicable when you run the mapping in the native environment or on the Spark engine.

For information about dynamic mappings, see the *Informatica Developer Mapping Guide*.

Refresh Schema

You can refresh the source or target schema at the runtime using control file when you enable a mapping to run dynamically. You can refresh the imported metadata before you run the dynamic mapping.

You can enable a mapping to run dynamically using the **At runtime, get data object columns from data source** option in the **Data Object** tab of the Read and Write transformations when you create a mapping.

When you add or override the metadata dynamically, you can include all the existing source and target objects in a single mapping and run the mapping. You do not have to change the source schema to update the data objects and mappings manually to incorporate all the new changes in the mapping.

You can use the mapping template rules to tune the behavior of the execution of such pipeline mapping.

When the Source or Target transformation contains updated ports such as changes in the port names, data types, precision, or scale, the Data Integration Service fetches the updated ports and runs the mapping dynamically. You must ensure that at least one of the column name in the source or target file is the same as before refreshing the schema to run the dynamic mapping successfully.

Even though the original order of the source or target ports in the table changes, the Data Integration Service displays the original order of the ports in the table when you refresh the schemas at runtime.

If there are more columns in the source file as compared to the target file, the Data Integration Service does not map the extra column to the target file and loads null data for all the unmapped columns in the target file.

If the Source transformation contains updated columns that do not match the Target transformation, the Data Integration Service does not link the new ports by default when you refresh the source or target schema. You must create a run-time link between the transformations to link ports at run time based on a parameter or link policy in the **Run-time Linking** tab.

For information about run-time linking, see the *Informatica Developer Mapping Guide*.

Mapping Flow

You can add all the Source transformation or transformation ports to the target dynamically when enable a mapping to run dynamically using the **Mapping Flow** option. You can then use the dynamic ports in the Write transformation.

When you select the **Mapping Flow** option, the Data Integration Service allows the Target transformation to override ports of the Write transformation with all the updated incoming ports from the pipeline mapping and loads the target file with the ports at runtime.

To enable a dynamic mapping using the **Mapping Flow** option, select the value of the **Columns defined by** property as **Mapping Flow** in the **Ports** tab in the Write transformation.

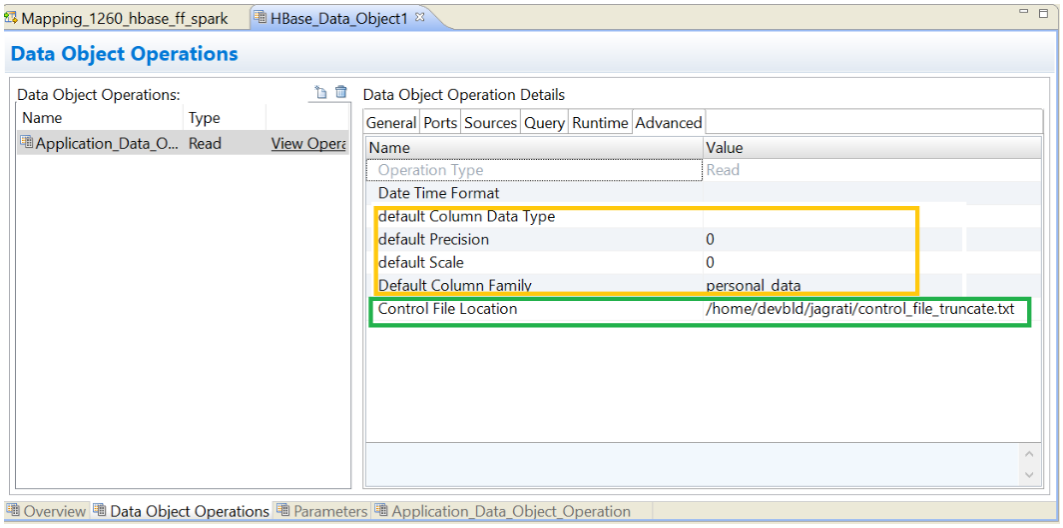
Dynamic Mapping with Control Files

You must use control file for dynamic mapping support in HBase at both source and target sides.

A control file is a simple text file with a field name, data type, precision, and scale. Each line in the control file contains one column of data.

To generate column names from a control file at run-time, select **From control file** for the **Generate Run-time Column Names** property in the Advanced properties of the HBase data object. You must also configure the Read transformation to get the column metadata at run time. You can use a control file to generate run-time column names for a Read transformation based on a HBase data object or to create a HBase data object.

You can configure a mapping (that is, to provide data type, precision, scale, and column family) to run dynamically from both UI **Run-time** tab or from a control file as shown in the following screen shot.



The control file has a new schema which is, column name, column data type, precision, scale, and column family.

The format of a control file is:

```
[SECTION1]
```

```
columnName[,[dataType],[precision],[scale][,{columnFamily=columnFamilyName}]]
```

The format defines the columns and its corresponding column family directly.

Note: The columnName and Header [SECTION1] is mandatory in the format of the control file.

Example of a control file:

```
[SECTION1]
name,string,20,0
city,,20,0
age,string,20,0,{columnFamily=personal_data}
workex
designation,{columnFamily=professional_data}
```

HBase Dynamic Mapping Example

Your organization has a large amount of data that keeps changing. Your organization needs to incorporate all the updated data in a short span of time. Create a dynamic mapping, where you can refresh the source schema dynamically to fetch the updated data. Add all the dynamic ports to the target to override the metadata of the existing ports.

1. Import a HBase data object and create read or write operations from the imported HBase data object.
2. Select a project or folder in the **Object Explorer** view.
3. Click **File** > **New** > **Mapping.**

   The **Mapping** dialog box appears.
4. Enter the name of the mapping in the **Name** field.
5. Click **Finish**.
6. Drag the source data object into a mapping.

   The **HBase Data Object Access** dialog box appears.
7. Select the **Read** option and click **OK**.
8. In the **Data Object** tab, select the **At runtime, get data object columns from data source** check box.
9. Drag the target data object into a mapping.

   The **HBase Data Object Acess** dialog box appears.
10. Select the **Write** option and click **OK**.
11. In the **Ports** tab, select the value of the **Columns defined by** as **Mapping Flow**.
12. Select all the source ports and add the ports to the target.
13. Save and run the mapping.

Filter Source Data

When you configure a mapping that reads data from an HBase source, you can enter a filter expression to filter records read from the source.

You can select the mapping and add the filter expression in the Query tab in the Properties view. You can use any comparison operators in the filter expression.

When you run the mapping, the Data Integration Service filters the source data based on the expressions.

Note: For numeric data types, the Data Integration Service applies the operators for positive values and not for negative values.

When you enter multiple filter expressions, the AND logical operator is applied between the expressions.

Note: If you use the not equal, less than, less than or equal to operators and some columns do not meet the filter condition, the Data Integration Service returns a Null value for these columns. If you use the equal, greater than, greater than or equal to operators and some columns do not meet the filter condition, the Data Integration Service does not return the rows associated with these columns.

Example

The following table lists the columns in the CF column family in an HBase table. There are rows that have c1 and c2 columns, rows that have at least one of the columns, and rows that have neither of the columns.

Row	Column Value
1	column=CF__c1, value=john
1	column=CF__c2, value=jane
2	column=CF__c1, value=jane
3	column=CF__c2, value=jdoe
4	column=CF__c8, value=adam

Create an HBase data object called Name and add it to an HBase mapping. Add the following filter expressions:

Name.CF__C1 = 'john' AND Name.CF__C2 = 'jane'

The Data Integration Service returns the following output because of the equal to operator in the filter expressions. The service does not return rows that contain null values.

ROW: 1

c1: john

c2: jane

However, the output is different when you add the following filter expressions:

Name.CF__C1 != 'john' AND Name.CF__C2 != 'jane'

The Data Integration Service returns rows that contain null values because of the not equal to operator in the filter expression.

Row 2

c1: jane

Row 3:

c2: jdoe

ROW: 4

c1: null

c2: null

Look up HBase Data

You can use an HBase data object read operation to look up data in an HBase resource. You add an HBase data object read operation as a Lookup transformation to a mapping.

You can then configure a lookup condition to look up data from the HBase resource.

Run the mapping on the Spark engine to look up data in an HBase resource.

Note: If an HBase lookup does not result in a match, it generates a row with NULL values for all columns. You can add an Expression transformation after the Lookup transformation to filter out NULL rows.

General Properties

The general properties display the name and description of the HBase data object read operation.

The following table describes the general properties that you can view and edit for an HBase lookup:

Property	Description
Name	Name of the HBase Lookup transformation.
Description	Description of the HBase Lookup transformation.
Physical Data Object	Name of the HBase data object read operation.
On multiple matches	Determines which row the HBase lookup returns when it finds multiple rows that match the lookup condition. You can select one of the following options: - Return first row - Return last row - Return any row - Return all rows - Report error

Ports Properties

The ports properties display the input ports from the source in the mapping to the HBase lookup. You can specify the ports to be available as output ports from the HBase lookup. The ports properties display the data types, precision, and scale of the source port.

The following table describes the ports properties:

Property	Description
Name	Name of the source port.
Type	Data type of the source port.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point of numeric values.
Output	Specify the ports that must be available as output ports from the HBase lookup.

Property	Description
Description	Description of the port.
Input Rules	A set of rules that filter the ports to include or exclude in the transformation based on port names or data type. Configure input rules when you define dynamic ports.

Run-time Properties

Set the run-time properties to enable and configure lookup caching. You must add the Lookup transformation to a mapping before you can configure run-time lookup properties.

Select the **Lookup caching enabled** check box to indicate whether the Data Integration Service must cache lookup values.

When you enable lookup caching, the Data Integration Service queries the lookup source once, caches the values, and looks up values in the cache. Caching the lookup values can increase performance on large lookup tables.

When you disable caching, each time a row passes into the transformation, the Data Integration Service issues a SELECT statement to the lookup source for lookup values.

Lookup Properties

You can specify the properties to look up an HBase resource. You can also parameterize the lookup condition.

The following table describes the lookup properties that you can specify for an HBase lookup:

Property	Description
Lookup Column	The name of the columns that you want to look up.
Operator	Operators that you can use to filter records. You can select one of the following operators: `=, !=, <=, >=, and`
Input Port	The input source port.

Configuring a Lookup Transformation

Configure a Lookup transformation to look up data in an HBase resource.

1. Open a mapping from the **Object Explorer** view.
2. From the **Object Explorer** view, drag an HBase data object read operation to the mapping editor.

   The **Add to Mapping** dialog box appears.
3. Select **Lookup** to add the data object read operation as a Lookup transformation to the mapping.
4. Select the HBase data object read operation and connect the lookup input ports and the lookup output ports.

5. In the **Properties** view, configure the following parameters:
   a. On the **General** tab, select the option that you want the Data Integration Service to return when it finds multiple rows that match the lookup condition.
   b. On the **Ports** tab, configure the output ports and input rules.
   c. On the **Run-time** tab, enable lookup caching if required.
   d. On the **Lookup** tab, configure the lookup condition.
6. When the mapping is valid, click **File** > **Save** to save the mapping to the Model repository.
7. Run the mapping in the on the Spark engine to look up data in an HBase resource.

Mapping Validation and Run-time Environments

You can validate and run mappings in the native environment or Hadoop environment.

The Data Integration Service validates whether the mapping can run in the selected environment.

You can deploy the mapping and run it in the selected environment. You can run standalone mappings or mappings that are a part of a workflow.

When you run a mapping in the native environment, the Data Integration Service runs the mapping from the Developer tool.

When you run a mapping on a Hadoop cluster, you can select the Blaze or Spark engine. The Data Integration Service pushes the mappings to the selected engine for processing.

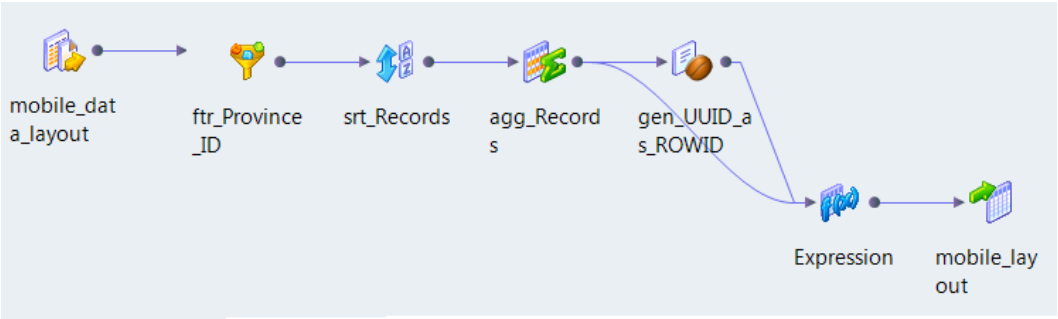
HBase Mapping Example

Your organization is a mobile service provider and it needs to load the data in WAP log files to HBase tables and generate multiple reports.

WAP log files can contain columns with information about the mobile users, internet usage, and data volume. On a single day, the WAP log files can be three to four billion rows of data and can be around two terabytes in size.

You can consolidate the data in the WAP log files that you receive through the day. You can then perform transformations based on your requirements.

The following figure shows the HBase mapping example:



You can use the following objects in an HBase mapping:

Flat File Data Object

The source for the mapping is a flat file data object that contains the data in a WAP log file.

Create a flat file data object and specify the WAP log file as the resource for the data object. Source columns in the flat file data object include Province ID, data volume, URL, and session duration. Configure the read properties of the data object.

Transformations

Add transformations to get aggregate data about the internet usage of the mobiles users in a particular province.

- The ftr_Province_ID Filter transformation filters the data in the log files based on the value you specify for the province ID column.
  The Data Integration Service returns the rows that meet the filter condition.
- The srt-Records Sorter transformation sorts the data in ascending order based on the province ID.
- The agg_Records Aggregator transformation collects statistics about internet usage and data volume of the mobile users for a particular province.
  Use the result of the Sorter transformation as an input to the Aggregator transformation. You can increase Aggregator transformation performance with the sorted input option.
- The gen_UUID_as_ROWID Java transformation generates a unique row key ID before you load the data to HBase tables.
  Each row in an HBase table has a unique row key ID. You can write the generated key value as the row key ID for each row in the HBase table
- The Expression transformation formats the data before you load it to the Hbase table.

HBase Data Object

The target of the mapping is an HBase data object. Specify the columns in the HBase table to which you want to write the data.

Create an HBase data object write operation to write data to the HBase table.

After you run the mapping, the Data Integration Service writes the transformed data to the HBase table. Analysts can run queries and perform real-time analysis of daily operations, statistics about gateway usage, and data volume based on the data in the HBase tables.

Rules and Guidelines for HBase Sources and Targets in a Mapping

Consider the following rules and guidelines for HBase sources and targets in a mapping:

- The control file of source and target can be located only in the domain location. Control file in HDFS location is not considered.
- You must verify that the target directory does not contain any folder or file with the same name as the target file name, to avoid deletion of files and folders.

APPENDIX A

Data Type Reference

This appendix includes the following topics:

Data Type Reference Overview

Informatica Developer uses the following data types in HBase mappings:

- HBase native data types. HBase data types appear in the physical data object column properties.

  Note: Although data is stored in binary format in HBase tables, you can specify the data type associated with a column when you create the HBase data object.

- Transformation data types. Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the Data Integration Service uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

When the Data Integration Service reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the Data Integration Service writes to a target, it converts the transformation data types to the comparable native data types.

HBase and Transformation Data Types

The following table lists the HBase data types that the Data Integration Service supports and the corresponding transformation data types:

HBase Data Type	Transformation Data Type	Range and Description
Binary	Binary	1 to 104,857,600 bytes. You can read and write data of Binary data type in a Hadoop environment. You can use the user-defined functions to transform the binary data.
BigDecimal	Decimal	Precision is 28, scale is 28.

HBase Data Type	Transformation Data Type	Range and Description
String	String	1 to 104,857,600 characters.
Short	Integer	-2,147,483,648 to 2,147,483,647.
Integer	Integer	-2,147,483,648 to 2,147,483,647.
Float	Double	Precision 15.
Long	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Double	Double	Precision 15
DateTime	Date/Time	Date and time formats are specified by using any of the Java date and time pattern strings. For example, The "EEE, d MMM yyyy HH:mm:ss z" pattern string is interpreted as Wed, 14 Dec 2013 12:08:56. The "yyyy.MM.dd HH:mm:ss z" pattern string is interpreted as 2013.12.14 12:08:56 PDT.

APPENDIX B

Glossary

column family

A group of multiple columns in an HBase table. HBase users can group columns based on appropriate logic to provide boundaries between the data. An HBase table has at least one column family, which is a collection of all the columns in the table. The column family name is the prefix for all the column names in that column family for unique identification.

HBase column

Basic unit for storing data in an HBase table.

HBase row

A row in an HBase table has one or more columns. Each row is uniquely identified by a row key. All rows are sorted lexicographically by their row key. In lexicographical sorting, each row key is compared on a binary level, byte by byte, from left to right with other row keys.

HBase table

An HBase table is made up of rows and columns similar to any database table.

INDEX