Informatica® PowerExchange for Microsoft Azure Blob Storage

10.4.1

User Guide

Informatica PowerExchange for Microsoft Azure Blob Storage User Guide
10.4.1
May 2020

Publication Date: 2021-08-17

Table of Contents

Preface 5

Informatica Resources 5

- Informatica Network 5
- Informatica Knowledge Base 5
- Informatica Documentation 5
- Informatica Product Availability Matrices 6
- Informatica Velocity 6
- Informatica Marketplace 6
- Informatica Global Customer Support 6

Chapter 1: Introduction to PowerExchange for Microsoft Azure Blob Storage 7

PowerExchange for Microsoft Azure Blob Storage Overview 7

PowerExchange for Microsoft Azure Blob Storage Example 7

Introduction to Microsoft Azure Blob Storage 8

Chapter 2: PowerExchange for Microsoft Azure Blob Storage Configuration 9

PowerExchange for Microsoft Azure Blob Storage Configuration Overview 9

Prerequisites 9

Java Heap Memory Configuration (Optional) 10

Configure Temporary Directory Location (Optional) 11

Chapter 3: Microsoft Azure Blob Storage Connections 12

Microsoft Azure Blob Storage Connection Overview 12

Microsoft Azure Blob Storage Connection Properties 12

Creating a Microsoft Azure Blob Storage Connection 13

Chapter 4: Microsoft Azure Blob Storage Data Objects 15

Microsoft Azure Blob Storage Data Objects Overview 15

Data Compression in Microsoft Azure Blob Storage Sources and Targets 16

- Configuring Lzo Compression Format 17

Microsoft Azure Blob Storage Data Object Properties 18

Microsoft Azure Blob Storage Data Object Read Operation Properties 18

- Directory Source in Microsoft Azure Blob Storage Sources 19
- Reading File Names for Source Objects 20
- Reading Files without Headers 22
- Schema Properties 22

Microsoft Azure Blob Storage Data Object Write Operation Properties 24

- Schema Properties 25

Creating a Microsoft Azure Blob Storage Data Object 28

Creating a Data Object Operation. 29
Creating a Microsoft Azure Blob Storage Target. 30

Chapter 5: Microsoft Azure Blob Storage Mappings. 31

Microsoft Azure Blob Storage Mappings Overview. 31
Mapping Validation and Run-time Environments. 31
Microsoft Azure Blob Storage Dynamic Mapping Overview. 32
    Refresh Schema. 32
    Mapping Flow. 33
Microsoft Azure Blob Storage Dynamic Mapping Example. 33

Chapter 6: Data Type Reference. 35

Data Type Reference Overview. 35
Microsoft Azure Blob Storage and Transformation Data Types. 36
Flat File and Transformation Data Types. 36
Avro Data Types and Transformation Data Types. 37
JSON Data Types and Transformation Data Types. 39
Parquet Data Types and Transformation Data Types. 39
Rules and Guidelines for Data Types. 41

Index. 43

Preface

Use the *Informatica® PowerExchange® for Microsoft Azure Blob Storage User Guide* to learn how to read from or write to Microsoft Azure Blob Storage by using the Developer tool. Learn to create a connection, develop and run mappings and dynamic mappings in the native environment and in the Hadoop and Databricks 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 Microsoft Azure Blob Storage

This chapter includes the following topics:

- PowerExchange for Microsoft Azure Blob Storage Overview, 7
- PowerExchange for Microsoft Azure Blob Storage Example, 7
- Introduction to Microsoft Azure Blob Storage, 8

PowerExchange for Microsoft Azure Blob Storage Overview

You can use PowerExchange for Microsoft Azure Blob Storage to connect to Microsoft Azure Blob Storage from Informatica.

Use PowerExchange for Microsoft Azure Blob Storage to read data from or write data to Microsoft Azure Blob Storage. You can read and write for secure transfer-enabled storage accounts. Use PowerExchange for Microsoft Azure Blob Storage to read delimited files and the industry-standard file formats, such as Avro, Parquet, and JSON files. You can read and write hierarchical data present in the Avro, Parquet, and JSON files. In addition to the industry-standard file formats, you can also read from intelligent structure sources.

You can read and write the gzip compressed `.csv` files only in the native environment. Create a Microsoft Azure Blob Storage connection to read or write Microsoft Azure Blob Storage data into a Microsoft Azure Blob Storage data object. When you use Microsoft Azure Blob Storage objects in mappings, you must configure properties specific to Microsoft Azure Blob Storage. You can validate and run mappings in the native and non-native environments.

You can use Microsoft Azure Blob Storage sources and targets in dynamic mappings.

PowerExchange for Microsoft Azure Blob Storage Example

You work in sales operations and want to score leads to drive higher sales for your organization. You need to bring in leads from Salesforce to Microsoft Azure Blob Storage. You can score leads for sales readiness in

Microsoft Azure Machine Learning, and then load the lead scores back into Salesforce. You can keep data up to date with the latest leads and lead scores by scheduling a workflow to run on a regular basis.

You have leads in Salesforce with data such as the contact information, industry, company size, and marketing information.

You configure a mapping to insert leads from Salesforce to Microsoft Azure Blob Storage. Use Microsoft Azure Machine Learning to score the leads, and then create another mapping to load the lead scores into Salesforce.

You create a workflow so that the tasks run serially promising leads and increase efficiency.

Introduction to Microsoft Azure Blob Storage

Microsoft Azure Blob Storage is a cloud-storage solution that stores unstructured data in the cloud as objects or blobs. Microsoft Azure Blob Storage can store text or binary data of any type, such as a document, media files, or application installer. Microsoft Azure Blob Storage is referred to as object storage.

Blobs are files of any type and size, and are organized into containers in Microsoft Azure Storage. You can access delimited files that are append blobs or block blobs with Microsoft Azure Blob Storage connections.

CHAPTER 2

PowerExchange for Microsoft Azure Blob Storage Configuration

This chapter includes the following topics:

- PowerExchange for Microsoft Azure Blob Storage Configuration Overview, 9
- Prerequisites, 9
- Java Heap Memory Configuration (Optional), 10
- Configure Temporary Directory Location (Optional), 11

PowerExchange for Microsoft Azure Blob Storage Configuration Overview

PowerExchange for Microsoft Azure Blob Storage installs with the Informatica services and clients.

Prerequisites

To successfully preview data from a local complex file or run a mapping in the native environment, you must configure the INFA_PARSER_HOME property for the Data Integration Service in Informatica Administrator. Perform the following steps to configure the INFA_PARSER_HOME property:

- Log in to Informatica Administrator.
- Click the Data Integration Service and then click the **Processes** tab on the right pane.
- Click **Edit** in the **Environment Variables** section.
- Click **New** to add an environment variable.
- Enter the name of the environment variable as **INFA_PARSER_HOME**.
- Set the value of the environment variable to the absolute path of the Cloudera CDH 6.1 directory on the machine that runs the Data Integration Service. For example:
  INFA_PARSER_HOME = *<Informatica installation directory>*/services/shared/hadoop/CDH_6.1

  Verify that the version of the Hadoop distribution directory that you define in the INFA_PARSER_HOME property is the same as the version you defined in the cluster configuration.

Configure Databricks Connection Advanced Properties

Verify that a Databricks connection is created in the domain. If you want to read NULL values from or write NULL values to an Azure source, configure the following advanced properties in the Databricks connection:

- `infaspark.flatfile.reader.nullValue=True`
- `infaspark.flatfile.writer.nullValue=True`

Configure Azure Blob Storage Access in Azure Databricks Cluster

Verify that a cluster configuration is created in the domain. Set your Azure Blob Storage account name and account key under **Spark Config** in your Databricks cluster configuration to access the Azure Blob Storage. Add "spark.hadoop" as a prefix to the Hadoop configuration key as shown in the following text:

```
spark.hadoop.fs.azure.account.key.<your-storage-account-name>.blob.core.windows.net
<your-storage-account-access-key>
```

Note: In case of multiple Azure Blob Storage accounts, you must configure the account name and account key for each of the Azure Blob Storage account.

Configure Azure Blob Storage SAS Access in Azure Databricks Cluster

Verify that a cluster configuration is created in the domain. Set your Azure Blob Storage account name and SAS token under **Spark Config** in your Databricks cluster configuration to access the Azure Blob Storage. Add "spark.hadoop" as a prefix to the Hadoop configuration key as shown in the following text:

```
spark.hadoop.fs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net
<sas-token-for-your-blob-account>
```

Configure access to secure transfer-enabled storage accounts

Verify that the **Secure transfer required** option in the **Configuration** tab in your Azure Blob Storage account is enabled. In addition, set the following custom property for the Data Integration Service:

```
SecureTransferRequired=True
```

After you configure the custom property, restart the Data Integration Service.

Java Heap Memory Configuration (Optional)

Configure the memory for the Java heap size in the node that runs the Data Integration Service.

1. In the Administrator tool, navigate to the Data Integration Service for which you want to change the Java heap size.
2. Edit the Custom Properties section in the Data Integration Service Properties.
3. To increase the heap memory size for a large dataset, define the following properties:

   ```
   ExecutionContextOptions.JVMMaxMemory = <size> MB
   ExecutionContextOptions.JVMMinMemory = <size> MB
   ```

   Where <size> is a valid heap size, such as 2048 MB.
4. Click Ok.
5. Restart the Data Integration Service.

Configure Temporary Directory Location (Optional)

Follow below steps to configure the temporary directory location in the node that runs the Data Integration Service.

1. In the Administrator tool, navigate to the Data Integration Service for which you want to change the temporary directory location.
2. Click the **Processes** tab.
3. Click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
4. Click **New** to add a new custom property.
5. Add the JVMOption custom property for the Data Integration Service and specify the value in the following format:

   ```
   -Djava.io.tmpdir=<required tmp directory location>
   ```

   For example,

   ```
   Property Name: JVMOption1
   Value: -Djava.io.tmpdir=/opt/Informatica/tmp/ZUDAP/
   ```
6. Click **Ok**.
7. Restart the Data Integration Service.

CHAPTER 3

Microsoft Azure Blob Storage Connections

This chapter includes the following topics:

- Microsoft Azure Blob Storage Connection Overview, 12
- Microsoft Azure Blob Storage Connection Properties, 12
- Creating a Microsoft Azure Blob Storage Connection, 13

Microsoft Azure Blob Storage Connection Overview

Microsoft Azure Blob Storage connection enables you to read data from or write data to Microsoft Azure Blob Storage.

You can use Microsoft Azure Blob Storage connections to create data objects and run mappings. The Developer tool uses the connection when you create a data object. The Data Integration Service uses the connection when you run mappings.

You can create an Microsoft Azure Blob Storage connection from the Developer tool or the Administrator tool. The Developer tool stores connections in the domain configuration repository. Create and manage connections in the connection preferences.

Microsoft Azure Blob Storage Connection Properties

Use a Microsoft Azure SQL Blob Storage connection to access a Microsoft Azure Blob Storage.

Note: The order of the connection properties might vary depending on the tool where you view them.

You can create and manage a Microsoft Azure Blob Storage connection in the Administrator tool or the Developer tool. The following table describes the Microsoft Azure Blob Storage connection properties:

Property	Description
Name	Name of the Microsoft Azure Blob Storage connection.
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.
Description	Description of the connection.
Location	The domain where you want to create the connection.
Type	Type of connection. Select Azure Blob Storage.

The **Connection Details** tab contains the connection attributes of the Microsoft Azure Blob Storage connection. The following table describes the connection attributes:

Property	Description
Account Name	Name of the Microsoft Azure Storage account.
Authorization Type	Authorization type. You can select any of the following authorization mechanisms: - Shared Key Authorization - Shared Access Signatures
Account Key	Microsoft Azure Storage access key. Applies when you select shared key authorization.
SAS Token	SAS URI with SAS token that you generate on Microsoft Azure portal for your account. Applies when you select shared access signature authorization type. **Note:** You must provide a valid SAS URI with a valid SAS token.
Container Name	The root container or sub-folders with the absolute path. **Note:** To import complex files, specify only the root container.
Endpoint Suffix	Type of Microsoft Azure end-points. You can select any of the following end-points: - `core.windows.net`: Default - `core.usgovcloudapi.net`: To select the US government Microsoft Azure end-points - `core.chinacloudapi.cn`: Not applicable

Creating a Microsoft Azure Blob Storage Connection

Before you create a Microsoft Azure Blob Storage data object, create a connection in the Developer tool.

1. Click **Window** > **Preferences**.
2. Select **Informatica** > **Connections**.

3. Expand the domain in the **Available Connections**.
4. Select the connection type **File System** > **Azure Blob Storage**, and click **Add**.
5. Enter a connection name and an optional description.
6. Enter an ID for the connection.
7. Select **Azure Blob Storage** as the connection type.
8. Click **Next**.
9. Configure the connection properties.
10. Click **Test Connection** to verify the connection to Microsoft Azure Blob Storage.
11. Click **Finish**.

CHAPTER 4

Microsoft Azure Blob Storage Data Objects

This chapter includes the following topics:

- Microsoft Azure Blob Storage Data Objects Overview, 15
- Data Compression in Microsoft Azure Blob Storage Sources and Targets, 16
- Microsoft Azure Blob Storage Data Object Properties, 18
- Microsoft Azure Blob Storage Data Object Read Operation Properties, 18
- Microsoft Azure Blob Storage Data Object Write Operation Properties, 24
- Creating a Microsoft Azure Blob Storage Data Object, 28
- Creating a Data Object Operation, 29
- Creating a Microsoft Azure Blob Storage Target, 30

Microsoft Azure Blob Storage Data Objects Overview

A Microsoft Azure Blob Storage data object is a physical data object that uses Microsoft Azure Blob Storage as a source or target. A Microsoft Azure Blob Storage data object represents the data in a Microsoft Azure Blob Storage file.

You can configure the data object read and write operation properties that determine how data can be read from Microsoft Azure Blob Storage sources and loaded to Microsoft Azure Blob Storage targets. You first create a connection to create a Microsoft Azure Blob Storage data object. When you create a data object, the read and write operations are created by default. You can modify the default read and write operations or create additional operations.

Data Compression in Microsoft Azure Blob Storage Sources and Targets

You can decompress data when you read data from Microsoft Azure Blob Storage and compress the data when you write data to Microsoft Azure Blob Storage.

Configure the compression format in the **Compression Format** option under the advanced source and target properties.

For the Flat resource type, select only the Gzip compression format in the native environment. The following table lists the compression formats for Avro, JSON, and Parquet resource types for a read operation:

Compression format	Avro File	Flat File	JSON File	Parquet File
None	Yes	Yes	No	Yes
Deflate*	Yes	N/A	Yes	No
Gzip	No	Yes	Yes	Yes
Bzip2	N/A	N/A	Yes	N/A
Lzo	N/A	N/A	No	Yes
Snappy*	Yes	N/A	Yes	Yes
**Select None to read the Deflate and Snappy file formats.*				

The following table lists the compression formats for Avro, JSON, and Parquet resource types for a write operation:

Compression format	Avro File	Flat File	JSON File	Parquet File
None	Yes	Yes	No	Yes
Deflate	Yes	N/A	Yes	No
Gzip	No	Yes	Yes	Yes
Bzip2	N/A	N/A	Yes	N/A
Lzo	N/A	N/A	No	Yes
Snappy	Yes	N/A	Yes	Yes

To read a compressed file from Microsoft Azure Blob Storage, the compressed file must have specific extensions. If the extensions used to read the compressed file are not valid, the Integration Service does not

process the file. The following table describes the extensions that are appended based on the compression format that you use:

Compression format	File Name Extension
Deflate	.deflate
Gzip	.GZ
Bzip2	.BZ2
Lzo	.LZO
Snappy	.snappy

Rules and guidelines for data compression

Consider the following guidelines when you read or write compressed files:

- To read multiple compressed files from a container, the compressed files must have same schema.
- You can read and write only primitive data types in the native environment.
- You can read and write both primitive and hierarchical data types in the non-native environment.
- You must enable the **Compressed Newly Created Blob** property to write a compressed file.
- You cannot append a compressed file to a target.
- To read and write Avro files compressed using Deflate to an Azure Blob Storage target, configure the following properties under Spark Config in your Databricks 5.1 cluster configuration:

  ```
  - spark.hadoop.avro.mapred.ignore.inputs.without.extension false
  - spark.sql.avro.compression.codec deflate
  - spark.sql.avro.deflate.level 5
  ```

Configuring Lzo Compression Format

To write the `.jar` files in the lzo compression format on the Spark and Databricks engines, you must copy the `.jar` files for the lzo compression on the machine on which the Data Integration Service runs.

For the Spark engine, perform the following steps to copy the `.jar` files from the distribution directory to the Data Integration Service:

1. Copy the `lzo.jar` file from the cluster to the following directories on the machine on which the Data Integration Service runs:
   `<Informatica installation directory>/<distribution>/infaLib`
2. Copy the lzo native binaries from the cluster to the following directory on the machine on which the Data Integration Service runs:
   `<Informatica installation directory>/<distribution>/lib/native`
3. In the Administrator Console, navigate to the Data Integration Service.
   The Data Integration Service page appears.
4. Click the **Processes** tab.
   The **Processes** page appears.
5. Click the pencil icon to edit the environment variables in the **Environment Variables** section.
   The **Edit Environment Variables** dialog box appears.

6. Click **New** to add a new environment variable.
   The **New Environment Variables** dialog box appears.
7. Enter the value of the **Name** field as `LD_LIBARY_PATH`.
8. Enter the following path in the **Value** field:
   `<infahome>/services/shared/bin:/<infahome>/services/shared/Hadoop/<distributionType>/lib/native`
9. Restart the Data Integration Service for changes to take effect.

For the Databricks engine, perform the following steps to copy the `.jar` files from the distribution directory to the Data Integration Service:

1. Copy the `lzo.jar` file from the cluster to the following directory on the machine on which the Data Integration Service runs:
   `<Informatica installation directory>/services/shared/hadoop/Databricks_<version>/runtimeLib`
2. Configure Spark Config in your Databricks cluster configuration to use the Lzo compression codec. The following snippet shows the sample configuration:

```
spark.hadoop.io.compression.codecs
"org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.DefaultCodec,com.hadoop.compression.lzo.LzoCodec,com.hadoop.compression.lzo.LzopCodec,org.apache.hadoop.io.compress.BZip2Codec
```

3. Restart the Data Integration Service for changes to take effect.

Note: For more information, see Microsoft Azure Databricks documentation.

Microsoft Azure Blob Storage Data Object Properties

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

The following table describes the properties that you configure for the Microsoft Azure Blob Storage data objects:

Property	Description
Name	Name of the Microsoft Azure Blob Storage data object.
Location	The project or folder in the Model Repository where you want to store the Microsoft Azure Blob Storage data object.
Connection	Name of the Microsoft Azure Blob Storage connection.

Microsoft Azure Blob Storage Data Object Read Operation Properties

Microsoft Azure Blob Storage data object read operation properties include advanced properties that apply to the Microsoft Azure Blob Storage data object.

The Developer tool displays advanced properties for the Microsoft Azure Blob Storage data object operation in the **Advanced** view.

The following table describes the advanced properties that you can configure for a Microsoft Azure Blob Storage data object read operation:

Property	Description
Number of concurrent connections to Blob Store	The number of concurrent connections to Blob Store to download files. Default is 4.
Compression Format	Decompresses data when you read data from Microsoft Azure Blob Storage. You can decompress the data in the following formats: - **None**. Select **None** to decompress files with the deflate and snappy formats. - **Bzip2** - **Gzip** - **Lzo** Default is None. You can read files that use the Bzip2 and Lzo compression formats in the non-native environments.
Source Type	Select the type of source from which you want to read data. You can select the following source types: - File - Directory Default is **File**. Applicable to the Spark mode. The directory read is not applicable to the binary file type.
Blob Name Override	Overrides the file name.
Blob Container Override	Overrides the default container name. You can specify the container name or sub-folders with an absolute or a relative path. **Note:** You must not use only slash '\' in the relative path to avoid creating folders without any names.

Directory Source in Microsoft Azure Blob Storage Sources

You can select the type of source from which you want to read data.

You can select the following type of sources from the **Source Type** option under the advanced properties for a Microsoft Azure Blob Storage data object read operation:

- File
- Directory

Note: The content type of the blob must be `application/x-gzip` to read compressed flat files for both File and Directory source types.

Use the following rules and guidelines to select **Directory** as the source type:

- The directory read is not applicable to the binary file type.
- All the source files in the directory must contain the same metadata.
- All the files must have data in the same format. For example, delimiters, header fields, and escape characters must be same.
- All the files under a specified directory are parsed. The files under subdirectories are not parsed.

- The connector does not perform any validation if there are multiple blob formats in the directory you select and might result into errors.

Reading File Names for Source Objects

A FileName port is a string port with a default precision of 1024 characters.

When you import a Microsoft Azure Blob Storage Source object, the Data Integration Service creates a FileName port in the imported data object. The FileName port stores the name of the file from which the Data Integration Service reads the data at run-time.

For example, a directory contains a number of files and each file contains multiple records that you want to read. You select the directory as source type in the Microsoft Azure Blob Storage Data Object Read Operation Properties. When you run the mapping, the Data Integration Service reads each record and stores the respective file name in the FileName port.

The FileName port is applicable to the following file formats in the native environment:

- Flat file
- Avro (excluding hierarchical data types)
- Binary
- JSON (excluding hierarchical data types)
- Parquet (excluding hierarchical data types)

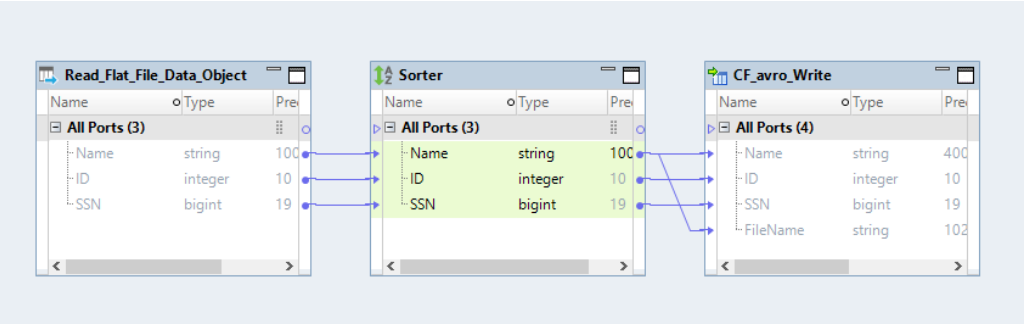
Rules and Guidelines for Using FileName Port

Use the following rules and guidelines when you use the FileName data in the FileName port:

- Do not use a colon (:) and forward slash (/) character in the file name data of the FileName port of the source or target object to run a mapping.
- If you connect the FileName port to the target, empty zero KB files are created in the target folder.
- Do not connect FileName port to a FileName port because the FileName port in the source might contain colon (:) and forward slash (/) characters.
- In the Native environment, use the Sorter transformation to sort the source port that you want to map to the FileName port of the Target transformation. After you sort the source port, map the port of the Sorter transformation to the FileName port of the Target transformation. The Data Integration Service creates only one file for each value with the same name. If you do not use the Sorter transformation, the Data Integration Service creates multiple files for each value with the same name.

  For example, create a mapping in the Native environment or on the Spark engine to read or write an Avro file using the FileName port.

  The following image shows the sorter transformation mapping:



If you want to Map the following source port name to the FileName port of the Target transformation and write the data to an Avro target file `target1`:

Name	ID	SSN
Anna	1	1
John	4	4
Smith	4	4
John	5	5
Anna	2	2

Add a Sorter transformation to sort the source port and map the source port to the port of the Sorter transformation. Then, map the port of the Sorter transformation to the FileName port of the Target transformation. The Data Integration Service creates the following directories and single file per thread within the directories:

```
target1.avro=Anna
```

In this directory, the Data Integration Service creates a file with the following values: `1,1,1,2,2,2`.

```
target1.avro=John
```

In this directory, the Data Integration Service creates a file with the following values: `4,4,4,5,5,5`.

```
target1.avro=Smith
```

In this directory, the Data Integration Service creates a file with the following values: `4,4,4`.

If you do not add a Sorter transformation, the Data Integration Service creates the following directories and multiple files within the directories:

```
target1.avro=Anna
```

In this directory, the Data Integration Service creates two part files with the following values: `1,1,1` and `2,2,2`.

```
target1.avro=John
```

In this directory, the Data Integration Service creates two files with the following values: `4,4,4` and `5,5,5`.

```
target1.avro=Smith
```

In this directory, the Data Integration Service creates one file with the following values: `4,4,4`.

Reading Files without Headers

You can read flat files that do not have headers and write data to a target.

You can select the following type of sources from the **Source Type** option :

1. Import a sample flat file object that has headers.
2. Enable column projection under **Schema Properties** of the data object.
3. Select `Flat` in schema format.
4. Select the appropriate schema.
5. Uncheck **Import column names from the first line** under **Column Format: Delimited** section.
6. Specify the actual flat file object name that does not have a header in **Blob Name Override** in advance properties.

Schema Properties

The Developer tool displays schema properties for intelligent structure model, Avro, JSON, and Parquet complex file sources in the **Data Object Operations Details** window.

The following table describes the schema properties that you configure for the complex file sources:

Property	Description
Column Name	Displays the name of the column.
Column Type	Displays the format of the column.
Enable Column Projection	Displays the column details of the complex files sources.
Schema Format	Displays the schema format that you selected while creating the complex file data object. You can change the schema format. You can also parameterize the schema format. You can select one of the following options: - Flat - Avro - JSON - Parquet - Intelligent Structure Model You can select the JSON file format when you run a mapping only on the Spark or Databricks Spark engine. You can change the complex file format without losing the column metadata even after you configure the schema properties for another complex file format. **Note:** You can switch from one schema format to another only once. If you change the schema format more than once, you might lose the original datatypes.

Property	Description
Schema	Displays the schema associated with the complex file. You can select a different schema. You can parameterize the schema or the schema path. To parameterize the schema path, obtain the path from the server. To parameterize schema or schema path for flat files, provide the schema in the same format in which the flat file is imported in the Developer Client. When you use Refresh Schema for the source or target in a mapping and also, parameterize the schema, the parameterized schema takes precedence over the refresh schema. **Note:** If you disable the column projection, the schema associated with the complex file is removed. If you want to associate schema again with the complex file, enable the column projection and select schema.
Schema Properties	Applicable only to flat files.
Column Mapping	Displays the mapping between input and output ports. **Note:** If you disable the column projection, the mapping between input and output ports is removed. If you want to map the input and output ports, enable the column projection and click **Select Schema** to associate a schema to the complex file.

Flat File Schema Properties

You can configure format properties for a flat file that is delimited.

The following table describes the file format and column format properties that you configure for a flat file:

Property	Description
Maximum row to preview	Number of rows to show in data preview. Default is 0.
Delimiters	Character used to separate columns of data. Default is comma. If you enter a delimiter that is the same as the escape character or the text qualifier, you might receive unexpected results.
Text Qualifier	Quote character that defines the boundaries of text strings. Default is double quotes. If you select a quote character, the Developer tool ignores delimiters within pairs of quotes.
Import Column Names From First Line	If selected, the Developer tool uses data in the first row for column names. Select this option if column names appear in the first row. Default is true.
Row Delimiter	You must select the default value, `\012 LF (\n)`.
Escape Character	Character immediately preceding a column delimiter character embedded in an unquoted string, or immediately preceding the quote character in a quoted string. When you specify an escape character, the Data Integration Service reads the delimiter character as a regular character. Default is backslash (\).
Retain Escape Character in Data	Not applicable.

Note: If you update the flat file format properties during the data object import and want to see the updated format properties in Data Preview, you must parse the flat file again by selecting the **Schema** property.

Parameterize Column Format Properties

You can parameterize the column format properties for flat files in a parameter file. The following table describes property strings that you can use in a parameter file:

Property	Value
maxRowsToPreview	A positive integer value. Default is 0.
delimiter	Specify the octal code for the character. Preface the octal code with a backslash (\). Specify the following values for the given delimiters: - `\011 TAB` for Tab - `;` for semicolon - `,` for comma - `\040 SP` for space Default is comma.
textQualifier	Specify the following string values: - SINGLE_QUOTES for ' - DOUBLE_QUOTES for " - NO_QUOTES Default is double quotes.
importColumnFromFirstLine	True or false. Default is true.
rowDelimiter	Default value, `\012 LF (\n)`.
escapeCharacter	A string value. Default is `\`.

A Sample JSON Parameter File

```
{ "maxRowsToPreview": 10, "delimiter": ";", "textQualifier": "SINGLE_QUOTES",
"importColumnFromFirstLine":true, "escapeCharacter" : "-" }
```

Microsoft Azure Blob Storage Data Object Write Operation Properties

Microsoft Azure Blob Storage data object write operation properties include advanced properties that apply to the Microsoft Azure Blob Storage data object.

The Developer tool displays advanced properties for the Microsoft Azure Blob Storage data object operation in the **Advanced** view.

The following table describes the advanced properties that you can configure for a Microsoft Azure Blob Storage data object write operation:

Property	Description
Number of concurrent connections to Blob Store	The number of concurrent connections to Blob Store to upload files. Default is 4.
Blob Name Override	Overrides the file name.
Blob Container Override	Overrides the default container name. You can specify the container name or sub-folders with an absolute or a relative path. **Note:** You must not use only slash '\' in the relative path to avoid creating folders without any names.
Compress newly created Blob	Compresses the newly created blob when set to True. Applicable to delimited files only in the native environment. You can compress Avro, JSON, and Parquet in the native and non-native environments.
Compression Format	Compresses data when you write data to Microsoft Azure Blob Storage. You can compress the data in the following formats: - **None** - **Bzip2** - **Deflate** - **Gzip** - **Lzo** - **Snappy** - **Zlib**: Not applicable Default is None. You can write files that use the Bzip2 and Lzo compression formats in the non-native environments.
Write Strategy	Override: overrides an existing blob. Append: appends to an existing blob. Append operation is applicable only to the append blob type. **Note:** Append is not applicable when you run a mapping in the non-native environments.
Blob Type	Writes data to a block blob or an append blob. You can write data to an append blob only for the flat format files in the native environment.
Target Schema Strategy	Not applicable

Schema Properties

The Developer tool displays schema properties for flat files and complex file targets in the **Data Object Operations Details** window.

The following table describes the schema properties that you configure for flat files and complex file targets:

Property	Description
Column Name	Displays the name of the column.
Column Type	Displays the format of the column.

Property	Description
Enable Column Projection	Displays the column details of the complex files sources.
Schema Format	Displays the schema format that you selected while creating the complex file data object. You can change the schema format. You can also parameterize the schema format. You can select one of the following options: - Flat - Avro - JSON - Parquet You can select the JSON file format when you run a mapping only on the Spark or Databricks Spark engine. You can change the complex file format without losing the column metadata even after you configure the column projection properties for another complex file format. **Note:** You can switch from one schema format to another only once. If you change the schema format more than once, you might lose the original datatypes.
Schema	Displays the schema associated with the complex file. You can select a different schema. You can parameterize the schema or the schema path. To parameterize the schema path, obtain the path from the server. When you use Refresh Schema for the source or target in a mapping and also, parameterize the schema, the parameterized schema takes precedence over the refresh schema. **Note:** If you disable the column projection, the schema associated with the complex file is removed. If you want to associate schema again with the complex file, enable the column projection and select schema.
Schema Properties	Applicable only to flat files.
Column Mapping	Displays the mapping between input and output ports. **Note:** If you disable the column projection, the mapping between input and output ports is removed. If you want to map the input and output ports, enable the column projection and click **Select Schema** to associate a schema to the complex file.

Flat File Schema Properties

You can configure format properties for a flat file that is delimited.

The following table describes the file format and column format properties that you configure for a flat file:

Property	Description
Maximum row to preview	Number of rows to show in data preview. Default is 0.
Delimiters	Character used to separate columns of data. Default is comma. If you enter a delimiter that is the same as the escape character or the text qualifier, you might receive unexpected results.
Text Qualifier	Quote character that defines the boundaries of text strings. Default is double quotes. If you select a quote character, the Developer tool ignores delimiters within pairs of quotes.

Property	Description
Import Column Names From First Line	Not applicable.
Row Delimiter	You must select the default value, `\012 LF (\n)`.
Escape Character	Character immediately preceding a column delimiter character embedded in an unquoted string, or immediately preceding the quote character in a quoted string. When you specify an escape character, the Data Integration Service reads the delimiter character as a regular character. Default is backslash (\).
Retain Escape Character in Data	Not applicable.

Note: If you update the flat file format properties during the data object import and want to see the updated format properties in Data Preview, you must parse the flat file again by selecting the **Schema** property.

Parameterize Column Format Properties

You can parameterize the column format properties for flat files in a parameter file. The following table describes property strings that you can use in a parameter file:

Property	Value
maxRowsToPreview	A positive integer value. Default is 0.
delimiter	Specify the octal code for the character. Preface the octal code with a backslash (\). Specify the following values for the given delimiters: - `\011 TAB` for Tab - `;` for semicolon - `,` for comma - `\040 SP` for space Default is comma.
textQualifier	Specify the following string values: - SINGLE_QUOTES for ' - DOUBLE_QUOTES for " - NO_QUOTES Default is double quotes.
rowDelimiter	Default value, `\012 LF (\n)`.
escapeCharacter	A string value. Default is `\`.

A Sample JSON Parameter File

```
{ "maxRowsToPreview": 10, "delimiter": ";", "textQualifier": "SINGLE_QUOTES",
"escapeCharacter" : "-" }
```

Creating a Microsoft Azure Blob Storage Data Object

Create a Microsoft Azure Blob Storage data object to add to a mapping.

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

   The Azure Blob Storage Data Object dialog box appears.
4. Enter a name for the data object.
5. In the **Resource Format** list, select any of the following formats:
   - Intelligent Structure Model: to read any format that an intelligent structure parses.
   - Binary: to read any resource format.
   - Flat: to read and write delimited resources.
   - Avro: to read and write Avro resources.
   - Json: to read and write JSON resources.
   - Parquet: to read and write Parquet resources.

   Note: Intelligent structure model is supported only on the Spark engine. JSON is supported in the non-native environments. Avro and Parquet are supported in the native and non-native environments.
6. Click **Browse** next to the **Location** option and select the target project or folder.
7. Click **Browse** next to the **Connection** option and select the AzureBlob connection from which you want to import the Microsoft Azure Blob Storage object.
8. To add a resource, click **Add** next to the **Selected Resources** option.

   The Add Resource dialog box appears.
9. Navigate through the container structure and select the checkbox next to the Microsoft Azure Blob Storage object you want to add and click **OK**.

   You must select the object according to the selected resource format. To use an intelligent structure model, select the appropriate `.amodel` file.
10. Click **Finish** if you selected Intelligent Structure Model, Avro, Json, or Parquet file and skip the remaining steps. Click **Next** if you selected a flat file.
11. Applicable only to the delimited files. Choose **Sample Metadata File** .

    You can click Browse and navigate to the directory that contains the file.
12. Click **Next**.

13. Configure the format properties.

Property	Description
Delimiters	Character used to separate columns of data. Make sure that the delimiter you select is not a part of the data. If you enter a delimiter that is the same as the escape character or the text qualifier, you might receive unexpected results. Microsoft Azure Blob Storage reader and writer support Delimiters.
Text Qualifier	Quote character that defines the boundaries of text strings. Make sure that the text qualifier you select is not a part of the data unless it is escaped. If you select a quote character, the Developer tool ignores delimiters within pairs of quotes. Microsoft Azure Blob Storage reader supports Text Qualifier.
Import Column Names From First Line	If selected, the Developer tool uses data in the first row for column names. Select this option if column names appear in the first row. Do not select this option to read header-less files. Applicable only to the native environment.

14. Click **Next** to configure the column properties and edit the column attributes.
15. Click **Finish**.

    The data object appears under Data Objects in the project or folder in the Object Explorer view. The data object read and write operations are created by default when a data object is created.

    Note: When you create a Microsoft Azure Blob Storage data object, the value of the folder path is displayed incorrectly in the **Resources** tab.

Creating a Data Object Operation

You can create the data object read or write operation for a Microsoft Azure Blob Storage data objects. You can then add the Microsoft Azure Blob Storage data object operation to a mapping.

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 read or write operation.

   Note: Create a read operation for a data object with an intelligent structure model. You cannot use a write transformation for a data object with an intelligent structure model in a mapping.
5. Click **Add**.

   The Select Resources dialog box appears.
6. Select the Microsoft Azure Blob Storage data object for which you want to create the data object operation and click **OK**.

   Note: Select the input or output file.
7. Click **Finish**.

   The Developer tool creates the data object operation for the selected data object.

Creating a Microsoft Azure Blob Storage Target

You can create a Microsoft Azure Blob Storage target using the **Create Target** option.

1. Select a project or folder in the **Object Explorer** view.
2. Select a source or a transformation in the mapping.
3. Right-click the Source transformation and select **Create Target**.
   The **Create Target** dialog box appears.
4. Select **Others** and then select **AzureBlob** data object from the list in the **Data Object Type** section.
5. Click **OK**.
   The **New AzureBlob Data Object** dialog box appears.
6. Enter a name for the data object.
7. In the **Resource Format** list, select any of the following formats to create the target type:
   - Avro
   - Flat
   - JSON
   - Parquet
8. Click **Finish**.

The new target appears under the **Physical Data Objects** category in the project or folder in the **Object Explorer** view.

CHAPTER 5

Microsoft Azure Blob Storage Mappings

This chapter includes the following topics:

- Microsoft Azure Blob Storage Mappings Overview, 31
- Mapping Validation and Run-time Environments, 31
- Microsoft Azure Blob Storage Dynamic Mapping Overview, 32
- Microsoft Azure Blob Storage Dynamic Mapping Example, 33

Microsoft Azure Blob Storage Mappings Overview

After you create a Microsoft Azure Blob Storage data object operation, you can develop a mapping.

You can define the following objects in the mapping:

- A Read transformation of the Microsoft Azure Blob Storage data object to read data from Microsoft Azure Blob Storage.
- A Write transformation of the Microsoft Azure Blob Storage data object to write data to Microsoft Azure Blob Storage.

Validate and run the mapping. You can deploy the mapping and run it or add the mapping to a Mapping task in a workflow. You cannot define a Lookup transformation of the Microsoft Azure Blob Storage data object.

Mapping Validation and Run-time Environments

You can validate and run mappings in the native environment or in a non-native environment, such as Hadoop or Databricks.

When you validate a mapping, you can validate it against one or all of the engines. The Developer tool returns validation messages for each engine.

When you run a mapping, you can choose to run the mapping in the native environment or in a non-native environment, such as Hadoop or Databricks. Configure the run-time environment in the Developer tool to optimize mapping performance and process data that is greater than 10 terabytes. When you run mappings in the native environment, the Data Integration Service processes and runs the mapping. When you run

mappings in a non-native environment, the Data Integration Service pushes the processing to a compute cluster, such as Hadoop or Databricks.

You can run standalone mappings, mappings that are a part of a workflow in a non-native environment. When you select the Hadoop environment, the Data Integration Service pushes the mapping logic to the Spark engine.

When you select the Databricks environment, the Integration Service pushes the mapping logic to the Databricks Spark engine, the Apache Spark engine packaged for Databricks.

Microsoft Azure Blob Storage Dynamic Mapping Overview

You can use Microsoft Azure Blob Storage data objects as dynamic sources and targets in a mapping.

Use the Microsoft Azure Blob Storage dynamic mapping to accommodate changes to source, target, and transformation logics at run time. You can use a Microsoft Azure Blob Storage dynamic mapping to manage frequent schema or metadata changes or to reuse the mapping logic for data sources with different schemas. Configure rules, parameters, and general transformation properties to create the dynamic mapping.

If the data source for a source or 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 transformation before you run the mapping again. The Data Integration Service dynamically determine transformation ports, transformation logic in the ports, and the port links within the mapping.

There are the 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.

Note: Dynamic mapping is applicable when you run the mapping in the native environment, on the Spark engine, or on the Databricks Spark engine. When you create a dynamic mapping to read multiple files from a directory and override the directory, verify that the override directory contains a source file with the same name as the imported object. Else, the mapping fails.

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

Refresh Schema

You can refresh the source or target schema at the runtime 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 file changes, the Data Integration Service displays the original order of the ports in the file 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 and update the target schema manually. For information about run-time linking, see the *Informatica Developer Mapping Guide*.

Even though you delete the FileName port from the Source transformation, the Data Integration Service adds the FileName port when you refresh the source schema.

Note: When you refresh a schema of a flat file, the Data Integration Service writes all data types as String data types.

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.

The Data Integration Service creates the target files dynamically based on the metadata of the incoming ports from the pipeline mapping.

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.

Microsoft Azure Blob Storage 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 the Microsoft Azure Blob Storage read and write data objects.

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 data object into a mapping.

   The **AzureBlob 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 data object into a mapping.

   The **AzureBlob 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 incoming ports and add the ports to the target.
13. Save and run the mapping.

CHAPTER 6

Data Type Reference

This chapter includes the following topics:

- Data Type Reference Overview, 35
- Microsoft Azure Blob Storage and Transformation Data Types, 36
- Flat File and Transformation Data Types, 36
- Avro Data Types and Transformation Data Types, 37
- JSON Data Types and Transformation Data Types, 39
- Parquet Data Types and Transformation Data Types, 39
- Rules and Guidelines for Data Types, 41

Data Type Reference Overview

Informatica developer uses the following data types in Microsoft Azure Blob Storage mappings:

- Microsoft Azure Blob Storage native data types. Microsoft Azure Blob Storage data types appear in the physical data object column properties.
- 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 a source, 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.

Microsoft Azure Blob Storage and Transformation Data Types

The following table lists the Microsoft Azure Blob Storage data types that the Data Integration Service supports and the corresponding transformation data types:

Microsoft Azure Blob Storage Native Data Type	Transformation Data Type	Range and Description
Bigint	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Decimal	Decimal	Precision 15
Integer	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
String	String	1 to 104,857,600 characters

Flat File and Transformation Data Types

Flat file data types map to transformation data types that the Data Integration Service uses to move data across platforms.

The following table compares flat file data types to transformation data types:

Flat File Data type	Transformation Data type	Range
Bigint	Bigint	Precision of 19 digits, scale of 0
Number	Decimal	For transformations that support precision up to 38 digits, the precision is 1 to 38 digits, and the scale is 0 to 38. For transformations that support precision up to 28 digits, the precision is 1 to 28 digits, and the scale is 0 to 28. If you specify the precision greater than the maximum number of digits, the Data Integration Service converts decimal values to double in high precision mode. If the precision is greater than 15, the Data Integration Service converts decimal values to double in low-precision mode.
String	String	1 to 104,857,600 characters
Nstring	String	1 to 104,857,600 characters

Avro Data Types and Transformation Data Types

Avro data types map to transformation data types that the Data Integration Service uses to move data across platforms.

The following table compares the Avro data types that the Data Integration Service supports and the corresponding transformation data types:

Avro Data Type	Transformation Data Type	Range
Array	Array	Unlimited number of characters.
Boolean	Integer	TRUE (1) or FALSE (0).
Bytes	Binary	Precision 4000.
Date	Date/Time	January 1, 0001 to December 31, 9999.
Decimal	Decimal	Decimal value with declared precision and scale. Scale must be less than or equal to precision. For transformations that support precision up to 38 digits, the precision is 1 to 38 digits, and the scale is 0 to 38. For transformations that support precision up to 28 digits, the precision is 1 to 28 digits, and the scale is 0 to 28. If you specify the precision greater than the maximum number of digits, the Data Integration Service converts decimal values to double in high precision mode.
Double	Double	Precision 15.
Fixed	Binary	1 to 104,857,600 bytes.
Float	Double	Precision 15.
Int	Integer	-2,147,483,648 to 2,147,483,647 Precision 10 and scale 0.
Long	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Precision 19 and scale 0.
Map	Map	Unlimited number of characters.
Record	Struct	Unlimited number of characters.
String	String	1 to 104,857,600 characters.
Time	Date/Time	Time of the day. Precision to microsecond.

Avro Data Type	Transformation Data Type	Range
Timestamp	Date/Time	January 1, 0001 00:00:00 to December 31, 9999 23:59:59.997. Precision to microsecond.
Union	Corresponding data type in a union of ["primitive_type|complex_type", "null"] or ["null", "primitive_type|complex_type"].	Dependent on primitive or complex data type.

Avro Union Data Type

A union indicates that a field might have more than one data type. For example, a union might indicate that a field can be a string or a null. A union is represented as a JSON array containing the data types.
The Developer tool only interprets a union of ["primitive_type|complex_type", "null"] or ["null", "primitive_type|complex_type"]. The Avro data type converts to the corresponding transformation data type.

Avro Timestamp Data Type Support

The following table lists the Timestamp data type support for Avro file formats:

Timestamp Data type	Native	Spark
Timestamp_micros	Yes	Yes
Timestamp_millis	Yes	No
Time_millis	Yes	No
Time_micros	Yes	No

Unsupported Avro Data Types

The Developer tool does not support the following Avro data types:

- Enum
- Null
- Timestamp_tz

JSON Data Types and Transformation Data Types

JSON data types map to transformation data types that the Data Integration Service uses to move data across platforms.

The following table compares the JSON data types that the Data Integration Service supports and the corresponding transformation data types:

JSON	Transformation	Range
Array	Array	Unlimited number of characters.
Double	Double	Precision of 15 digits.
Integer	Integer	-2,147,483,648 to 2,147,483,647. Precision of 10, scale of 0.
Object	Struct	Unlimited number of characters.
String	String	1 to 104,857,600 characters.

Note: You can use JSON data types to read and write complex file objects in mappings that run on the Spark engine only.

Unsupported JSON Data Types

The Developer tool does not support the following JSON data types:

- Date
- Decimal
- Timestamp
- Enum
- Union

Parquet Data Types and Transformation Data Types

Parquet data types map to transformation data types that the Data Integration Service uses to move data across platforms.

The following table compares the Parquet data types that the Data Integration Service supports and the corresponding transformation data types:

Parquet	Transformation	Range
Binary	Binary	1 to 104,857,600 bytes
Binary (UTF8)	String	1 to 104,857,600 characters

Parquet	Transformation	Range
Boolean	Integer	-2,147,483,648 to 2,147,483,647 Precision of 10, scale of 0
Date	Date/Time	January 1, 0001 to December 31, 9999.
Decimal	Decimal	Decimal value with declared precision and scale. Scale must be less than or equal to precision. For transformations that support precision up to 38 digits, the precision is 1 to 38 digits, and the scale is 0 to 38. For transformations that support precision up to 28 digits, the precision is 1 to 28 digits, and the scale is 0 to 28. If you specify the precision greater than the maximum number of digits, the Data Integration Service converts decimal values to double in high precision mode.
Double	Double	Precision of 15 digits.
Float	Double	Precision of 15 digits.
Int32	Integer	-2,147,483,648 to 2,147,483,647 Precision of 10, scale of 0
Int64	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision of 19, scale of 0
Map	Map	Unlimited number of characters.
Struct	Struct	Unlimited number of characters.
Time	Date/Time	Time of the day. Precision to microsecond.
Timestamp	Date/Time	January 1, 0001 00:00:00 to December 31, 9999 23:59:59.997. Precision to microsecond.
group (LIST)	Array	Unlimited number of characters.

The Parquet schema that you specify to read or write a Parquet file must be in smaller case. Parquet does not support case-sensitive schema.

Parquet Timestamp Data Type Support

The following table lists the Timestamp data type support for Parquet file formats:

Timestamp Data type	Native	Spark
Timestamp_micros	Yes	No
Timestamp_millis	Yes	No
Time_millis	Yes	No
Time_micros	Yes	No
int96	Yes	Yes

Unsupported Parquet Data Types

The Developer tool does not support the following Parquet data types:

- Timestamp_nanos
- Time_nanos
- Timestamp_tz

Rules and Guidelines for Data Types

Consider the following rules and guidelines for data types:

- Avro data types support:
  - Date, Decimal, and Timestamp data types are applicable when you run a mapping in the native environment or on the Spark engine in Cloudera CDH 6.1 distribution.
  - Time data type is applicable when you run a mapping in the native environment in Cloudera CDH 6.1 distribution.
- JSON data types support:
  - For PowerExchange for Microsoft Azure Data Lake Storage Gen2, you can read and write complex file objects in JSON format in mappings that run in the native environment, Spark engine, and Databricks Spark engine.
    For other file-based adapters, you can read and write complex file objects in JSON format in mappings that run on the Spark engine only.
- Parquet data types support:
  - When you set the `-DINFA_HADOOP_DIST_DIR=hadoop\<Distro>` option in the `developerCore.ini` file and import a Parquet file, the format of the imported metadata differs based on the distribution. For Cloudera CDP 7.1, the metadata is imported as string and for other supported distributions, the metadata is imported as UTF8.
  - Date, Time, and Timestamp data types till microseconds are applicable when you run a mapping in the native environment , Blaze, and Spark engine in the Cloudera CDH 6.1, Hortonworks HDP 3.1, Azure HDInsight HDI 4.0, and Cloudera CDP 7.1 distributions.

- Date,Time_Millis and Timestamp_Millis data types are applicable when you run a mapping in the native environment or Spark engine in the Amazon Elastic MapReduce (EMR) 5.23, MapR 6.1, and Hortonworks HDP 2.6 distributions.
- Decimal data types are applicable when you run a mapping in the native environment and Spark engine in Cloudera CDH 6.1, Cloudera CDH 6.3, Hortonworks HDP 2.6, Hortonworks HDP 3.1, Amazon EMR 5.20, Amazon EMR 5.23, MapR 6.1, Google Dataproc 1.4, and Azure HDInsight HDI 4.0 distributions.
- Date, Time, Timestamp, and Decimal data types are applicable when you run a mapping on the Databricks Spark engine.
- When you run a mapping and use Date data type that does not have a time value, the Data Integration Service adds the time value, based on the time zone, to the date in the target.
  For example, Date data type used in the source:

  ```
  1980-01-09
  ```

  Value generated in the target:

  ```
  1980-01-09 00:00:00
  ```
- When you run a mapping in the native environment and use Time data type in the source, the Data Integration Service writes incorrect date value to the target.
  For example, Time data type used in the source:

  ```
  1980-01-09 06:56:01.365235000
  ```

  Incorrect Date value is generated in the target:

  ```
  1899-12-31 06:56:01.365235000
  ```
- When you run a mapping in the native environment and use Date data type in the source, the Data Integration Service writes incorrect time value to the target.
  For example, Date data type used in the source:

  ```
  1980-01-09 00:00:00
  ```

  Incorrect Time value generated in the target:

  ```
  1980-01-09 05:30:00
  ```

- To run a mapping that reads and writes Date, Time, Timestamp, and Decimal data types, update the `-DINFA_HADOOP_DIST_DIR` option to the `developerCore.ini` file. The `developerCore.ini` file is located in the following directory:
  `<Client installation directory>\clients\DeveloperClient\`

  Add the following path to the `developerCore.ini` file:

  ```
  -DINFA_HADOOP_DIST_DIR=hadoop\CDH_6.1
  ```

- To use precision up to 38 digits for Decimal data type in the native environment, set the `EnableSDKDecimal38` custom property to `true` for the Data Integration Service. The `EnableSDKDecimal38` custom property is applicable to all file-based PowerExchange adapters except PowerExchange for HDFS.

INDEX

A

Avro data types
  transformation data types 37

C

Configure Temporary Directory Location 11
configuring
  lzo compression format 17
Connection
  details 12
  properties 12
Create
  Data objects 12
  storage connection 13
create target
  Microsoft Azure Blob Storage 30

D

Data
  objects 15
data compression
  sources and targets 16
Data object
  operation 29
  properties 18
  storage 28
Data types
  description 36
  range 36
directory source
  Microsoft Azure Blob Storage sources 19

E

Example
  Microsoft Azure Blob Storage 7

F

Flat file
  transformation data types 36

H

headerless files 22

J

java heap size 10
JSON data types
  transformation data types 39

M

mapping flow
  dynamic mapping 33
mappings
  run 12
Microsoft Azure
  Blob Storage 7
Microsoft Azure Blob Storage
  dynamic mapping 32
Microsoft Azure Blob Storage dynamic mapping
  example 33
Microsoft Azure Blob Storage mappings
  overview 31
Microsoft Azure Blob Storage run-time environment
  description 31
Microsoft Azure Blob Storage validation environment
  description 31

P

Parquet data types
  transformation data types 39
PowerExchange 7
PowerExchange for Microsoft Azure Blob Storage configuration
  overview 9
Prerequisites 9

R

read operation
  flat file schema properties 23
read operation properties
  schema properties 22
refresh schema
  dynamic mapping 32
rules and guidelines
  FileName port 20

S

Storage connection
  overview 12
storage solution
  cloud 8

U

unstructured data
  blobs 8
  objects 8

W

working with FileName port 20
write operation
  flat file schema properties 26
write operation properties
  schema properties 25