

Spectrum™ Technology Platform

Version 12.0 SP1

Enterprise Data Integration Guide



Table of Contents

1 - Introduction

Enterprise Data Management Architecture	5
The Star Schema Data Warehouse Design	8

2 - Connecting to Data Sources and Data Warehouses

Data Source connections	14
Connection Types	18
Deleting a Connection	66

3 - Populating the Data Warehouse

Preparing Your Data	68
Populating a Time Dimension Table	69
Populating a Dimension Table	70
Populating a Fact Table	72
Adding a Time Stamp to Records in a Data Warehouse	76

4 - Updating the Data Warehouse

Defining a Data Warehouse Update Schedule	79
Updating a Fact Table	80
Using a Global Cache for Queries	84
Using a Local Cache for Queries	85

5 - Stages Reference

Call Stored Procedure	88
DB Change Data Reader	90
DB Loader	93
Field Parser	102
Field Combiner	105
Field Selector	106
Generate Time Dimension	107
Query Cache	114
Query DB	122
Query NoSQL DB	125
Read From DB	128
Read From File	136
Read from Hadoop Sequence File	153
Read From Hive File	157
Read from HL7 File	160
Read from NoSQL DB	172
Read from SAP	176
Read from Spreadsheet	181
Read from Variable Format File	183
Read From XML	197
SQL Command	204
Transposer	213
Unique ID Generator	217
Write to Cache	225
Write to DB	227
Write to File	232
Write to Hadoop Sequence File	249
Write to Hive File	253
Write to NoSQL DB	259
Write to Spreadsheet	263
Write to Variable Format File	266
Write to XML	276
Date and Number Patterns	283

6 - Configurations

Oracle LogMiner Configurations 291

7 - Optimizing Performance

Determining an Optimum Fetch Size 294

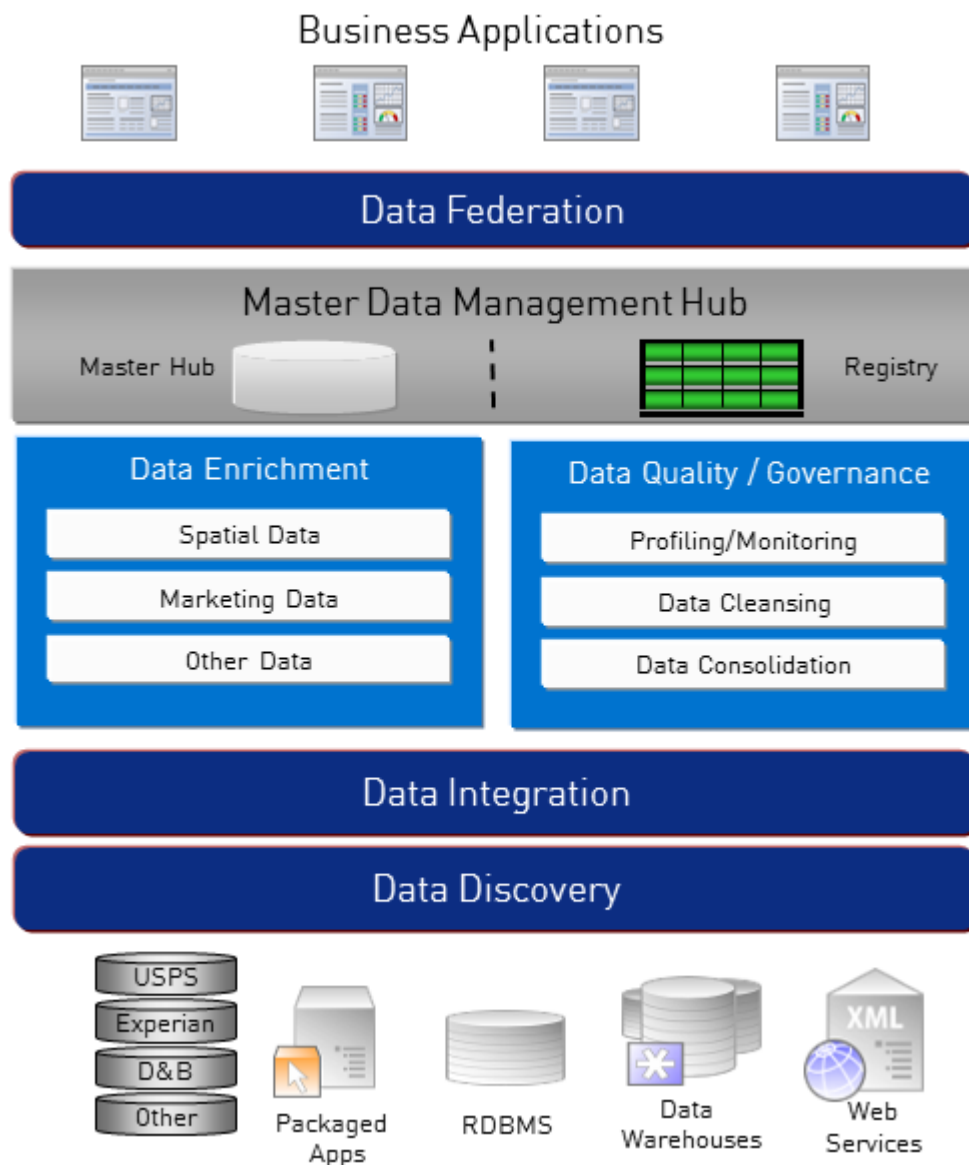
1 - Introduction

In this section

Enterprise Data Management Architecture	5
The Star Schema Data Warehouse Design	8

Enterprise Data Management Architecture

With Spectrum™ Technology Platform, you can build a comprehensive enterprise data management process, or you can use it as a more targeted solution. The following diagram illustrates a complete solution that takes data from its source, through data enrichment and data quality processes, feeding a master data management hub which makes a single view of the data available to multiple business applications.



Data Discovery

Data discovery is the process of scanning your data resources to get a complete inventory of your data landscape. Spectrum™ Technology Platform can scan structured data, unstructured data, and semi-structured data using a wide array of data profiling techniques. The results of the scan are used to automatically generate a library of documentation describing your company's data assets and to create a metadata repository. This documentation and accompanying metadata repository provide the insight you need before beginning data integration, data quality, data governance, or master data management projects.

For more information on the Spectrum™ Technology Platform Data Discovery Module, contact your account executive.

Data Integration

Once you have an inventory of your data landscape, you need to consider how you will access the data you need to manage. Spectrum™ Technology Platform can connect to data in multiple sources either directly or through integration with your existing data access technologies. It supports batch and real time data integration capabilities for a variety of business needs including data warehousing, data quality, systems integration, and migration. Spectrum™ Technology Platform can access data in RDBMS databases, data warehouses, XML files, flat files, and more. Spectrum™ Technology Platform supports SQL queries with complex joins and aggregations and provides a visual query development tool. In addition, Spectrum™ Technology Platform can access data over REST and SOAP web services.

Spectrum™ Technology Platform can trigger batch processing based on the appearance of one or more source files in a specified folder. This "hot folder" trigger is useful for monitoring FTP uploads and processing them as they occur.

Some of these data integration capabilities require a license for the Enterprise Data Integration Module. For more information, contact your account executive.

Finally, Spectrum™ Technology Platform can integrate with packaged applications such as SAP and Siebel.

Data Quality/Governance

Data quality and data governance processes check your data for duplicate records, inconsistent information, and inaccurate information.

Duplicate matching identifies potential duplicate records or relationships between records, whether the data is name and address in nature or any other type of customer information. Spectrum™ Technology Platform allows you to specify a consistent set of business match rules using boolean matching methods, scoring methods, thresholds, algorithms and weights to determine if a group of records contains duplicates. Spectrum™ Technology Platform supports extensive customization so you can tailor the rules to the unique needs of your business.

Once duplicate records have been identified, you may wish to consolidate records. Spectrum™ Technology Platform allows you to specify how to link or merge duplicate records so you can create the most accurate and complete record from any collection of customer information. For example,

a single best-of-breed record can be built from all of the records in a household. The Advanced Matching Module is used to identify duplicates and eliminate them.

Data quality processes also standardize your data. Standardization is a critical process because standardized data elements are necessary to achieve the highest possible results for matching and identifying relationships between records. While several modules perform standardization of one type or another, the Spectrum™ Technology Platform Data Normalization module provides the most comprehensive set of standardization features. In addition, the Universal Name module provides specific data quality features for handling personal name and business name data.

Standardized data is not necessarily accurate data. Spectrum™ Technology Platform can compare your data to known, up-to-date reference data for correctness. The sources used for this process may include regulatory bodies such as the U.S. Postal Service, third-party data providers such as Experian or D&B, or your company's internal reference sources, such as accounting data. Spectrum™ Technology Platform is particularly strong in address data validation. It can validate or standardize addresses in 250 countries and territories around the world. There are two modules that perform address validation: the Address Now Module and the Universal Addressing Module.

To determine which one is right for you, discuss your needs with your account executive.

While Spectrum™ Technology Platform can automatically handle a wide range of data quality issues, there are some situations where a manual review by a data steward is appropriate. To support this, the Business Steward Module provides a way to specify the rules that will trigger a manual review, and it provides a web-based tool for reviewing exception records. It includes integrated access to third-party tools such as Bing maps and Experian data to aid data stewards in the review and resolution process.

Data Enrichment

Data enrichment processes augment your data with additional information. Enrichment can be based on spatial data, marketing data, or data from other sources that you wish to use to add additional detail to your data. For example, if you have a database of customer addresses, you could geocode the address to determine the latitude/longitude coordinates of the address and store those coordinates as part of the record. Your customer data could then be used to perform a variety of spatial calculations, such as finding the bank branch nearest the customer. Spectrum™ Technology Platform allows you to enrich your data with a variety of information, including geocoding (with the Enterprise Geocoding Module), tax jurisdiction assignment (with the Enterprise Tax Module), geospatial calculations (with the Location Intelligence Module), and driving and walking directions between points (with the Enterprise Routing Module).

Master Data Management Hub

The Master Data Management (MDM) hub allows for rapid modeling of entities and their complex relationships across roles, processes and interactions. It provides built-in social network analysis capabilities to help you understand influencers, predict churn, detect non-obvious relationships and fraudulent patterns, and provide recommendations.

Spectrum™ Technology Platform supports two approaches to the MDM hub. In the master hub approach, the data is maintained in a single MDM database and applications access the data from

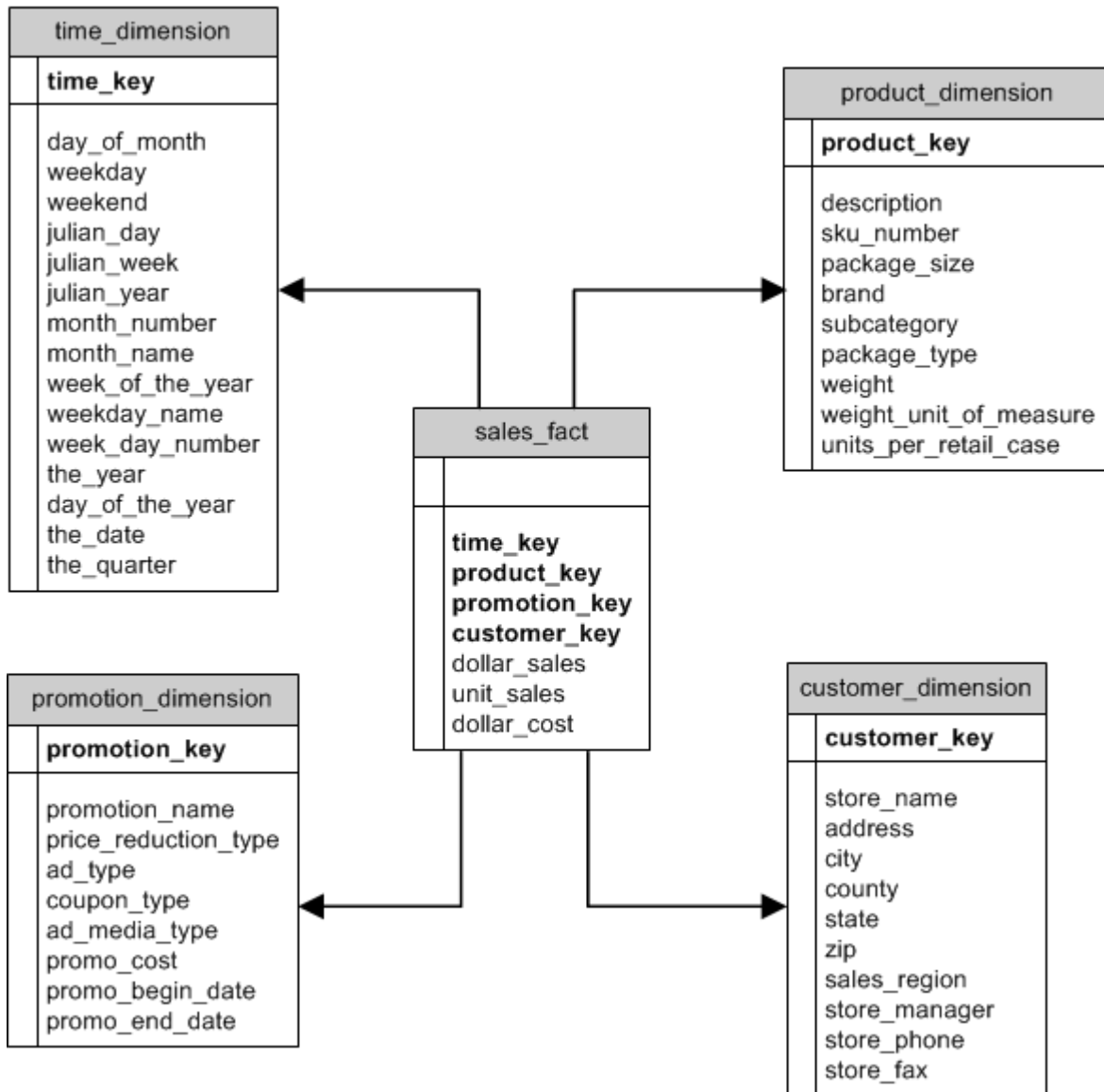
the MDM database. In the registry approach, the data is maintained in each business application and the MDM hub registry contains keys which are used to find related records. For example, a customer's record may exist in an order entry database and a customer support database. The MDM registry would contain a single key which could be used to access the customer data in both places.

The Data Hub Module provides MDM capabilities.

The Star Schema Data Warehouse Design

Spectrum™ Technology Platform supports the creation and maintenance of data warehouses that use a star schema design. In a star schema, data is stored as either facts, which are specific descriptions of an event, or dimensional attributes, which are descriptions of the facts in the fact table. Facts change regularly and dimensions change slowly or never.

The following illustration shows the design of a star schema:



This illustration shows the main characteristics of a star schema: a fact table, dimension tables, and joins.

Fact Table

Fact tables are the central tables in the star schema of your data warehouse. Fact tables usually contain numeric or quantitative information (called measures) that describe a specific event. For example, if you have a data warehouse that you use to generate a report on company revenue, you would have `dollar_sales`, and `dollar_cost` as columns within your fact table, as shown in the illustration above. Typically, facts are continuously valued and additive. "Continuously valued" means that the fact is a numeric measurement that has a value every time it is measured. "Additive" means that the fact can be summarized through addition.

Fact tables also contain a set of columns that form a concatenated, or composite key. Each column of the concatenated key is a foreign key drawn from a dimension table's primary key. For example, in the above illustration the fact table contains a column `product_key` which associates the fact with a specific product in the `product_dimension` table.

The level of detail in a fact table is called the grain. Every row in the fact table must be recorded to the same level of detail. In the diagram above, the measurements in the fact table are daily totals of sales in dollars, sales in units, and cost in dollars of each product sold. The grain is daily. Each record in the fact table represents the total sales of a specific product in a retail store on one day. Each new combination of product, store, or day generates a different record in the fact table.

Fact tables are populated with data extracted from a data source. The data source can be an OLTP system or a data warehouse. Spectrum™ Technology Platform takes a snapshot of the source data on a regular schedule and moves the data to the data warehouse, usually at the same time every day, week or month.

A star schema can have multiple fact tables. Use a schema with multiple fact tables to separate sets of measurements that share a common subset of dimension tables, or to track measurements with different grains.

Dimension Table

Dimension tables store data that describe the information in the fact table. For example, if `sales_total` differed one month from the next you would look to the dimensions to tell you why. The same dimension table can be used with different fact tables.

Dimension tables have attributes and a single part primary key that joins the dimension table to the fact table. Attributes are the columns in the dimension table. The single part primary key allows you to quickly browse a single dimension table. Browsing a dimension table can help determine the best way to query the fact table.

Time dimension tables are necessary for accurate time-based calculations because you sometimes cannot easily extract the necessary date data from the records. For example, the following records are in a sales database. Note that there are time gaps between records. For example, there is no record for the day 1/4/2012.

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/5/2012	Red Shirt	\$5.00
1/7/2012	Red Shirt	\$15.00

If you query these records and calculate the average sales per day, the answer would be \$10.00 ($\$30 / 3$ records). However, this is incorrect because the three records actually span a period of five

days. If you have a time dimension table with a record for each day, you could join that table with the above table to get this:

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/4/2012		
1/5/2012	Red Shirt	\$5.00
1/6/2012		
1/7/2012	Red Shirt	\$15.00

Calculating the average sales per day using these records, you would get the correct answer: \$6.00 (\$30 / 5 days).

In addition, you could account for arbitrary time attributes such as holidays, weekends, and quarters in your calculation. For example, if 1/6/2012 happened to be a holiday and you were only interested in average sales per workday then the answer would be \$7.50.

Joins

Joins define relationships between a fact table and dimension tables in the star schema. The primary key in the dimension table is the foreign key in the fact table. The fact table must contain a primary key value from each dimension table. The reference from the foreign key to the primary key is the mechanism for verifying values between the two tables. Join relationships of this type ensure the referential integrity of a data warehouse. Referential integrity must be maintained to ensure valid query results.

Each record in a dimension table can describe many records in the fact table, making the join cardinality of dimension tables to fact tables one-to-many.

In the illustration above, `product_key` is the primary key in the `product_dimension` table and the foreign key in the `sales_fact` table. This join represents the relationship between the company's products and its sales.

Advantages of a Star Schema

A well-designed schema allows you to quickly understand, navigate and analyze large multidimensional data sets. The main advantages of star schemas in a decision support environment are:

Query Performance

Queries run faster against a star schema database than an OLTP system because the star schema has fewer tables and clear join paths. In a star schema design, dimensions are linked through the central fact table. Dimensions are linked with each other through one join path intersecting the fact table. This design feature enforces accurate and consistent query results.

Load Performance and Administration

The star schema structure reduces the time required to load large batches of data into a database. By defining facts and dimensions and separating them into different tables, the impact of a load operation is reduced. Dimension tables can be populated once and occasionally refreshed. New facts can be added regularly and selectively by appending records to a fact table.

Built-in Referential Integrity

A star schema is designed to enforce referential integrity of loaded data. Referential integrity is enforced by the use of primary and foreign keys. Primary keys in dimension tables become foreign keys in fact tables to link each record across dimension and fact tables.

Efficient Navigation Through Data

Navigating through data is efficient because dimensions are joined through fact tables. These joins are significant because they represent fundamental relationships of real business processes. You can browse a single dimension table in order to select attribute values to construct an efficient query.

2 - Connecting to Data Sources and Data Warehouses

In this section

Data Source connections	14
Connection Types	18
Deleting a Connection	66

Data Source connections

A data source is a database, file server, cloud service, or other source of data that you want to process through Spectrum™ Technology Platform. Spectrum™ Technology Platform can connect to over 20 types of data sources.

To connect Spectrum™ Technology Platform to a data source, use Management Console to define the connection. For example, if you want to read data from an XML file into a dataflow, and the XML file is located on a remote file server, you would have to define a connection to the file server before you can define the input XML file in a dataflow. Similarly, if you want to write dataflow output to a database, you must first define the database as an external resource.

Note: If you want to read from or write to data located in a file on the Spectrum™ Technology Platform server itself there is no need to define a connection.

Compression Support for Cloud File Servers

The file servers Amazon S3, Google cloud storage, and MS Azure Blobstore support the compressed formats `gzip (.gz)` and `zip (.zip)`.

The Spectrum™ Technology Platform handles the compression and decompression of the files written to and read from the file servers.

Note: You can use the same file server to handle both normal reads and writes of files as well as compression and decompression of files.

Reading a Compressed Format File

While reading a file from the server, its compression format is derived from the metadata key property `Content-Encoding` received from the server.

Writing a Compressed Format File

While writing a file to a server, mention the required compression format: `.gz` or `.zip`. The file is compressed based on the specified compression extension.

The metadata key property `Content-Encoding` is also set according to the selected compression format. This property value is passed to the cloud file server while writing the file to it.

Supported Entities and Operations

Spectrum™ Technology Platform supports these entities and operations for each connection type:

Connection Type Table Properties

Marketo

The entities are of the following types:

1. Entity
2. Entity Update

Note: Entity Update is a virtual table used for Update on Lead entity. For example, **Merge_Leads** should be used for merging different Marketo Leads.

MS Dynamics CRM

The entities are of the following types:

1. User Owned
 2. Organization Owned
 3. Business Owned
 4. None
-

Netsuite

The entities are of the types:

- Standard Records
- Custom Records
- Joins
- Saved Searches

On viewing the schema of the created Baseview, for each entity type, the resultant schema is displayed.

For example, for a Saved Search record, the schema of the search result is displayed. For a Join record, the schema of the join's result is displayed.

Note: In a NetSuite connection table, the primary key column is `internalId`.

SAP

1. The entity columns are of two types:

- **Native:** Columns with native datatypes are displayed with their respective datatypes.
- **Custom-defined:** Columns with custom-defined datatypes are displayed with a blank datatype.

To deploy a Virtual Data Source derived from an SAP connection, ensure its Metaviews and Baseviews include only such entities whose columns are of native datatypes. If the Baseviews and Metaviews have entities of custom-defined datatypes, the Virtual Data Source cannot be deployed.

Siebel

The business components are displayed in the format `Business Object.Business Component`.

Connection Type Table Properties

Splunk	Supported Operations	LIKE, ORDER BY, LIMIT, IN, BETWEEN, !=, <=, >=, <, >, multiple AND/OR operators	
	Supported Functions	String Functions	upper, lower, length, len, ltrim, rtrim, substring, max, min
		Mathematical Functions	abs, ceil, exp, floor, sqrt, round

Note: For all other query operations, use the Splunk `search` column as explained below.

Spectrum™ Technology Platform provides a column `search` in the Splunk table using which you can look up the required data in the Splunk connection.

While executing a `select` query on the `SplunkTable`, use the `search` column in the `where` clause in any of the below scenarios:

1. To include such search criteria which cannot be specified using ANSI SQL syntax.
2. To include such Splunk-specific search criteria which cannot be included as part of the main SQL query.

For example, the below query looks for such a `_raw` value which contains the key `opp` with the value `ACC`.

```
select "_raw" from SplunkTable where "search"='search opp=ACC'
```


Connection Type Table Properties

SuccessFactors	<p>The entities are of two types:</p> <ol style="list-style-type: none"> 1. Entity: Represents a table representing a business entity. 2. Join: Represents a mapping between any two <i>Entity</i> type tables: a parent table and any of its child tables. <p>Note: Links are not present between tables in the Baseview schema derived from a SuccessFactors connection. This is because foreign keys are not present in SuccessFactors tables, and joins between tables are indicated by <i>Join</i> type tables in the Spectrum™ Technology Platform.</p>
----------------	--

The features of Join tables are:

1. The name of a *Join* table indicates the two *Entity* tables which have been mapped together.
2. Each record of a *Join* table contains the primary key from the parent entity and the columns of the respective child entity. Thus the particular parent entity's primary key is mapped to the child entity's details.

For example, `User#HR` is a *Join* table in which `User` is the parent entity and `Hr` is the child entity. This join represents all the users and their respective HR representatives. The join table `User#HR`, therefore, has the parent table `User`'s primary key `UserId`, which is mapped to the columns of the child table `HR`, like `hr_userId`, `hr_username`, `hr_email`, and so on.

3. In case of *Join* tables, the `insert` and `update` functions work like the `upsert` function. This is because *Join* tables are not actual entities in SuccessFactors, but are mappings between entities and their navigation properties or child tables.

To `insert/update` any *Join* table, the parent entity is updated, while in child table a new record is inserted or the existing record is updated corresponding to the parent record.

Note: While updating, the mapping between a parent and a child is modified. It is also possible to modify the individual attributes of the child as required.

SugarCRM	<p>Supported operations:</p> <p><code>LIKE</code> (its operation is limited to picking up options starting with the specified value, such as in the statement <code>WHERE name LIKE 's%'</code>, which picks up all the names starting with the alphabet S.), <code>IS NULL</code>, <code>IS NOT NULL</code>, <code>IN</code>, <code>NOT IN</code>, <code>></code>, <code>>=</code>, <code><</code>, <code><=</code>, <code>=</code>, <code><></code>, <code>AND</code>, <code>OR</code></p>
----------	---

Oracle Eloqua	<p>Supported Operations</p> <p>Three entity types are supported:</p> <ol style="list-style-type: none"> 1. Entity: Denotes a table representing business entity. 2. Activity: Denotes a table representing a business entity where data is generated based on some activity. 3. Custom Entity: Denotes entities that are used as part of special operations provided with the Connector.
---------------	---


Connection Types

Connecting to Amazon

Connecting to Amazon DynamoDB

In order for Spectrum™ Technology Platform to access data in Amazon DynamoDB you must define a connection to Amazon DynamoDB using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Amazon DynamoDB.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.


5. In the **Type** field, choose **Amazon DynamoDB**.
6. In the **Access Key ID** field, enter the 20-character alpha-numeric sequence provided to you to access your Amazon AWS account.
7. In the **Secret Access Key** field, enter the 40-character key needed to authenticate the connection.
8. In the **Region** field, select the region of the Amazon AWS account.
9. To test the connection, click **Test**.
10. Click **Save**.

Amazon DynamoDB Limitations

1. Hierarchical data types like lists, sets and maps are interpreted as String data types. This is because these data types are not supported.
2. Null values in a DynamoDB data source are interpreted as empty column values.
3. The `count` aggregate function is not supported in query on Model Store.

Connecting to Amazon S3

In order for Spectrum™ Technology Platform to access data in Amazon S3 you must define a connection to Amazon S3 using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Amazon S3.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Cloud**.
6. In the **Cloud service** field, choose **AmazonS3**.
7. In the **Bucket name** field, enter the bucket name as defined in your Amazon S3 cloud service. This is the bucket where Spectrum™ Technology Platform will read and write files.
8. Enter your access key and secret key assigned to you by Amazon..
9. In the **Storage Type**, field select the level of redundancy that you want to allow for data storage.

Standard	The default level of redundancy provided by Amazon S3.
Reduced redundancy	Stores non-critical and easily-reproducible data at lower levels of redundancy. This provides fairly reliable storage at a lower cost.

10. In the **Encryption** section, select the encryption method for the data. You can select server side encryption, client side encryption, or both.

Server side key The data is encrypted and decrypted at the server side. Your data is transmitted in plain text to the Amazon cloud service where it is encrypted and stored. On retrieval, the data is decrypted by the Amazon cloud service then transmitted in plain text to your system.

You have two options for specifying the key:

- **AWS managed:** The key is automatically generated by the Amazon S3 cloud service.
- **Customer provided:** Enter the key to be used by the Amazon S3 cloud service to encrypt and decrypt the data on the server side.

Client side key The data is encrypted and decrypted at the client side. The data is encrypted locally on your client system then transmitted to the Amazon S3 cloud storage. On retrieval, the data is transmitted back in an encrypted format to your system and is decrypted on the client system.

Client side key: Enter the key to be used by your client system to encrypt and decrypt the data.

If you select both **Server side key** and **Client side key**, encryption and decryption is performed at both server and client sides. Data is first encrypted with your client side key and transmitted in an encrypted format to Amazon, where it is again encrypted with the server side key and stored. On retrieval, Amazon first decrypts the data with the server side key, transmitting the data in an encrypted format to your system, where it is finally decrypted with the client side key.

Note: To use the encryption feature of Amazon S3 cloud, you need to install the Amazon S3 Security JAR files. For more information, see [Using Amazon S3 Cloud Encryption](#) on page 20.

For more information about Amazon S3 encryption features, see:

docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html

11. If you want to set access permissions, in the **Permissions** section, click .

The three kinds of Grantees are:

Everyone	Every one else other than Authenticated Users and Log Delivery group.
AuthenticatedUsers	For users who are logged into Amazon.
LogDelivery	For users who write activity logs in a user-specified Bucket, if Bucket Logging is enabled.

For each Grantee, select the desired permissions:

Open/Download	Allow the user to download the file.
View	Allow the user to view the current permissions on the file.
Edit	Allow the user to modify and set the permissions on the file.

12. To test the connection, click **Test**.
13. Click **Save**.

[Using Amazon S3 Cloud Encryption](#)

To use the encryption security feature of the Amazon S3 cloud service, you need to download security JAR files and place them on the Spectrum™ Technology Platform server. Using encryption is optional.

1. Go to the download site.

For Windows and Linux platforms using Java 7, the JAR files can be downloaded from:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

For AIX platforms using Java 7, the JAR files can be downloaded from:

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk>

2. Download these two JAR files:

- local_policy.jar
- US_export_policy.jar

3. Place the JAR files in the location:


```
SpectrumFolder\Pitney Bowes\Spectrum\java64\jre\lib\security
```

4. Restart the server.

Connecting to Amazon SimpleDB

In order for Spectrum™ Technology Platform to access data in Amazon SimpleDB you must define a connection to Amazon SimpleDB using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Amazon SimpleDB.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Amazon SimpleDB**.
6. In the **Access Key ID** field, enter the 20-character alpha-numeric sequence provided to you to access your Amazon AWS account.
7. In the **Secret Access Key** field, enter the 40-character key needed to authenticate the connection.
8. To test the connection, click **Test**.
9. Click **Save**.

Amazon SimpleDB Limitations

Write Limitation

In the Write to DB stage, the write mode **Update** is not available when writing to an Amazon SimpleDB table. The **Insert** option handles both insert and update operations. It differentiates between an insert and an update using the unique value of the `ItemName` column which is present in all Amazon SimpleDB tables.

Reason: An update query requires a Primary Key for each record of the table to be updated, which is not supported by Amazon SimpleDB databases.


Read Limitation

The aggregate functions `SUM` and `AVG` are not supported while executing queries on Model Store.

Connecting to Apache Cassandra

In order for Spectrum™ Technology Platform to access data in a Cassandra database, you must define a connection to the Cassandra database using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, the Cassandra database.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.


5. In the **Type** field, choose **Apache Cassandra**.
6. In the **Host** field, enter the machine name or the IP on which the Apache Cassandra database is installed.
7. In the **Keyspace** field, enter the name of the keyspace of the data center you wish to access.
8. In the **Port** field, enter the port on which the Apache Cassandra database is configured.
9. Enter the user name and password to use to authenticate to the Cassandra database.
10. In the **Consistency Level** field, select how consistent data rows must be across replica nodes for a successful data transaction. This can be at least one, or all, or a combination of available nodes.
11. In the **Fetch Size**, enter the number of resultset rows you wish to fetch on each read transaction.
12. To test the connection, click **Test**.
13. Click **Save**.

Apache Cassandra Limitation

The `count` aggregate function is not supported in query on Model Store.

Connecting to Azure Cloud

In order for Spectrum™ Technology Platform to access data in Microsoft Azure, you must define a connection to Microsoft Azure using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Microsoft Azure.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.


5. In the **Type** field, choose **Cloud**.
6. In the **Cloud service** field, choose **AzureBlobStorage**.
7. In the **Protocol** field select whether you want the connection between Azure and Spectrum™ Technology Platform to use HTTP or HTTPS.
8. In the **Account Name** field, enter the name of your Azure storage account.
9. In the **Access Key** field, enter the access key to your Azure account.
10. To test the cloud connection, click **Test**.
11. Click **Save**.

Connecting to a Flat File

Connecting to a Delimited Flat File

To add a new Delimited Flat File connection navigate to **Connections > Flat File**, and select the **Record Type** as **Delimited**. Enter the file's access details and content type details to allow the Data Federation Module to read it correctly.

Note: This connection is for use in the Metadata Insights module.

1. Go to **Connections > Flat File**.
2. By default, the screen opens in the create mode. Otherwise, click  to add a new Flat File connection.
3. Enter a **Connection Name** for the Flat File data connection.
4. Enter the **File Path** by clicking **Browse** and selecting the directory of the file.
5. Select the **Character Encoding** of the flat file from the drop-down.


6. Select the **Record Type** as **Delimited**.
7. In **Field Delimiter**, select the expected separator between any two fields of a file record.
8. Select the **Text Qualifier (optional)**, if any, that encloses the field values of a file record.
9. In **Line Separator**, the value `Default` is selected, indicating that the expected line separator depends on whether Spectrum™ Technology Platform is running on a Unix or Windows system.
10. To specify whether the first row of the file is a header row, shift the **First Row is Header Row** slider to either **Yes** or **No**.
11. To specify whether the data type of the various fields in any record of the file should be automatically detected, shift the **Detect data type from file** slider to either **Yes** or **No**.
12. To skip malformed records during file parsing, shift the **Skip Malformed Records** slider to **On**.
13. Click **Test**.
A message confirms the successful test of the connection.
14. Click **Save**.
A message confirms the successful creation of the connection.

In order to view a sample record fetched using the created Delimited Flat File connection, click **Preview** in the header bar. File records will be fetched and the Fields sorted according to the details provided by you.

Connecting to a Fixed Width Flat File

To add a new Fixed Width Flat File connection navigate to **Connections > Flat File** and select the **Record Type** as **Fixed Width**. Enter the file's access details and content type details to allow the Data Federation Module to read it correctly.

Note: This connection is for use in the Metadata Insights module.

1. Go to **Connections > Flat File**.
2. By default, the screen opens in the create mode. Otherwise, click  to add a new Flat File connection.
3. Enter a **Connection Name** for the Flat File data connection.
4. Enter the **File Path** by clicking **Browse** and selecting the directory of the file.
5. Select the **Character Encoding** of the flat file from the drop-down.
6. Select the **Record Type** as **Fixed Width**.
7. In the **Record Length** field, enter the total number of characters in a file record.

Repeat Step 8 to Step 13 to enter details of all fields expected in a file record.

8. Click **Add Field** to add a row for a field in a file record.
9. In the **Name** column, enter the name for the field value.
10. In the **Type** column, select the data type of the field value.
11. In the **Start Position** column, enter the position in the file record at which of the field value begins.

For the first field in a file record, the **Start Position** counting begins from 1.

12. In the **Length** field, enter the total number of characters the field covers, including the character at the **Start Position**.

The sum of the **Start Position** and **Length** values for any field should be less than or equal to the **Record Length**

If the File Record is:

```
01234Rob Smith29PitneyBowes
```

Record Length = 27

For the field 'Name':

Start Position = 6

Length = 9

```
Name = Rob Smith
```

13. Check the **Trim** checkbox if you wish to trim any white spaces at the beginning and/or end of a field value.
14. Click **Test**.
A message confirms the successful test of the connection.
15. Click **Save**.
A message confirms the successful creation of the connection.

In order to view a sample record fetched using the created Fixed Width Flat File connection, click **Preview** in the header bar. File records will be fetched and the Fields sorted according to the details provided by you.

Date Time Formats in a File Connection

While reading date and time values from files using a File Connection in Spectrum™ Technology Platform, the values need to adhere to certain specific date-time formats.

Accepted Date Time Formats

- Date: "yyyy-mm-dd"
- Datetime: "yyyy-mm-dd HH:mm:ss"
- Time: "HH:mm:ss"

These are as per standard date-time notations.

Delimited Files

If the **Detect type** feature is turned on while configuring the Delimited File Connection, then the date and time values in the file records, which adhere to the above formats, are automatically detected as Date type.

If a date-time value does not adhere to one of the accepted formats, the value is read as a String type value instead of a Date type value.

Fixed Width Files

For Fixed Width files, date type values are configured while creating the Fixed Width File Connection. Hence these values are read as Date type values, irrespective of whether they adhere to the accepted formats or not.

If the date-time value in a Fixed Width file does not adhere to the accepted formats, it needs to be handled using **Transformations** at the logical model creation stage by applying the below *Conversion* category function to the value:

```
parsedate(String date, String format)
```

In this, the *date* is the value received from the file, while the *format* is the date-time format in which the value is received from the file. This helps to parse the date-time value correctly.

For example, if the *date* = 23-Feb-2008, then the *format* = dd-*MMM*-*yyyy*.

Resulting Value Formats

While previewing data in a model store:

- If the value has been read as a date/time value, it is reflected in one of the accepted date/time formats in the preview.
- If the value has been read as a String value, it is reflected as it is in the preview.


Connecting to an FTP Server

In order for Spectrum™ Technology Platform to access files on an FTP server you must define a connection to the FTP server using Management Console. Once you do this, you can create dataflows in Enterprise Designer that can read data from, and write data to, files on the FTP server.

Before connecting to an FTP server, verify that the timeout settings on your FTP server are appropriate for the jobs that will use this connection. Depending on the design of a job, there may be periods of time when the connection is idle, which may cause the connection to time out. For example, you may have a dataflow with two Read from File stages connected to an Import To Hub stage. While the Import To Hub stage is reading records from one Read from File stage, the other will be idle, possibly causing its connection to the FTP server to time out. Consider setting the timeout value on your FTP server to 0 to prevent connections from timing out.

Note: The FTP server must be running in active connection mode. Passive connection mode is not supported.


1. Open Management Console.
2. Go to **Resources > Data Sources**.

3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.
5. In the **Type** field, choose **FTP**.
6. In the **User name** and **Password** fields, enter the credentials to use to authenticate to the FTP server. This is required only if the FTP server requires it.
7. In the **Host** field, enter the hostname or IP address of the FTP server.
8. In the **Port** field, enter the network port number the server uses for FTP.
9. Click **Test** to verify that the Spectrum™ Technology Platform server can connect to the FTP server.
10. Click **Save**.

Connecting to Google Cloud Storage

In order for Spectrum™ Technology Platform to access data in Google Cloud Storage, you must define a connection to Google Cloud Storage using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Google Cloud Storage.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.
5. In the **Type** field, choose **Cloud**.
6. In the **Cloud service** field, choose **GoogleCloudStorage**.
7. In the **Bucket name** field, enter the bucket name as defined in your Google cloud service. This is the bucket where Spectrum™ Technology Platform will read and write files.
8. Enter your the application name, service account, and private key file provided by Google.

Note: Ensure the private key file is present on the Spectrum™ Technology Platform server.
9. You can set access permissions in the **Permissions** section.

Manage your data and permission Allows the user to manage the data and permissions.

View your data Allows the user to view data.

Manage your data

Allows the user to manage data.


10. To test the connection, click **Test**.
11. Click **Save**.

For more information, see Google's [Service Account Authentication](#) documentation.

Connecting to Hadoop

In order for Spectrum™ Technology Platform to access data in Hadoop, you must define a connection to Hadoop using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Hadoop.

Attention: Spectrum™ Technology Platform does not support *Hadoop 2.x* for Kerberos on Windows platforms.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **HDFS**
6. In the **Host** field, enter the hostname or IP address of the NameNode in the HDFS cluster.
7. In the **Port** field, enter the network port number.
8. In **User**, select one of:

Server user	Choose this option if authentication is enabled in your HDFS cluster. This option will use the user credentials that the Spectrum™ Technology Platform server runs under to authenticate to HDFS.
User name	Choose this option if authentication is disabled in your HDFS cluster.
9. Check **Kerberos** if you wish to enable Kerberos authentication feature for this HDFS file server connection.
10. If you have opted to enable **Kerberos** authentication, then enter the path of the keytab file in the **Keytab file path** field.

Note: Ensure the key tab file is placed on the Spectrum™ Technology Platform server.

11. In the **Protocol** field, select one of:

WEBHDFS	Select this option if the HDFS cluster is running HDFS 1.0 or later. This protocol supports both read and write operations.
----------------	---

HFTP	Select this option if the HDFS cluster is running a version older than HDFS 1.0, or if your organization does not allow the WEBHDFS protocol. This protocol only supports the read operation.
HAR	Select this option to access Hadoop archive files. If you choose this option, specify the path to the archive file in the Path field. This protocol only supports the read operation.

12. Expand the **Advanced options**.



13. If you selected the WEBHDFS protocol, you can specify these advanced options as required:

Replication factor	Specifies how many data nodes to replicate each block to. For example, the default setting of 3 replicates each block to three different nodes in the cluster. The maximum replication factor is 1024.
Block size	Specifies the size of each block. HDFS breaks up a file into blocks of the size you specify here. For example, if you specify the default 64 MB, each file is broken up into 64 MB blocks. Each block is then replicated to the number of nodes in the cluster specified in the Replication factor field.
File permissions	Specifies the level of access to files written to the HDFS cluster by Spectrum™ Technology Platform. You can specify read and write permissions for each of these options:

Note: The *Execute* permission is not applicable to Spectrum™ Technology Platform.

User	This is the user specified above, either Server user or the user specified in the User name field.
Group	This refers to any group of which the user is a member. For example, if the user is john123, then Group permissions apply to any group of which john123 is a member.
Other	This refers to any other users as well as groups of which the specified user is not a member.

14. In the grid below the **File permissions** table, specify the server properties for Hadoop to ensure that the sorting and filtering features work as desired when the connection is used in a stage or activity. To add properties, do one of these:

- Click  and add the properties and their respective values in the **Property** and **Value** fields.
- Click  and upload your configuration XML file. The XML file should be similar to hdfs-site.xml, yarn-site.xml, or core-site.xml.

Note: The configuration file needs to be placed on the server.

This table describes the properties and their values, depending on the stage or activity that will use the Hadoop connection. The properties are also dependent on the Hadoop version used (*Hadoop 1.x* or *Hadoop 2.x*).

Stage or Activity using the HDFS Connection	Required Server Properties
---	----------------------------

- Stage **Read from Sequence File**
- Activity **Run Hadoop Pig**

Stage or Activity using the
HDFS Connection

Required Server Properties

Hadoop 1.x Parameters

fs.default.name

Specifies the node and port on which Hadoop runs.

For example,

`hdfs://152.144.226.224:9000`

mapred.job.tracker

Specifies the hostname or IP address, and port on which the MapReduce job tracker runs. If the host name is entered as local, then jobs are run as a single map and reduce task.

For example, `152.144.226.224:9001`

dfs.namenode.name.dir

Specifies where on the local files system a DFS name node should store the name table. If this is a comma-delimited list of directories, then the name table is replicated in all of the directories, for redundancy.

For example,

`file:/home/hduser/Data/namenode`

dfs.datanode.data.dir

Specifies where on the local file system a DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all the named directories that are usually on different devices. Directories that do not exist are ignored.

For example,

`file:/home/hduser/Data/datanode`

hadoop.tmp.dir

Specifies the base location for other temporary directories.

For example, `/home/hduser/Data/tmp`

Hadoop 2.x Parameters

Stage or Activity using the
HDFS Connection

Required Server Properties

fs.defaultFS

Specifies the node and port on which Hadoop runs.

For example,

```
hdfs://152.144.226.224:9000.
```

Note: For Spectrum versions 11.0 and earlier, the parameter name `fs.defaultfs` must be used. Note the case difference.

For versions 11 SP1 and onwards, both the names `fs.defaultfs` and `fs.defaultFS` are valid. We recommend using the parameter name `fs.defaultFS` Spectrum™ Technology Platform 11 SP1 onwards.

yarn.resourcemanager.resource-tracker.address

Specifies the hostname or IP-address of the Resource Manager.

For example, `152.144.226.224:8025`

yarn.resourcemanager.scheduler.address

Specifies the address of the Scheduler Interface.

For example, `152.144.226.224:8030`

yarn.resourcemanager.address

Specifies the address of the Applications Manager interface that is contained in the Resource Manager.

For example, `152.144.226.224:8041`

Stage or Activity using the
HDFS Connection

Required Server Properties

mapreduce.jobhistory.address

Specifies the host name or IP address, and port on which the MapReduce Job History Server is running.

For example, `152.144.226.224:10020`

mapreduce.application.classpath

Specifies the CLASSPATH for Map Reduce applications. This CLASSPATH denotes the location where classes related to Map Reduce applications are found.

Note: The entries should be comma separated.

For example

```
$HADOOP_CONF_DIR,  
$HADOOP_COMMON_HOME/share/hadoop/common/*,  
$HADOOP_COMMON_HOME/share/hadoop/common/lib/*,  
$HADOOP_HDFS_HOME/share/hadoop/hdfs/*,  
$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,  
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*,  
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*,  
$HADOOP_YARN_HOME/share/hadoop/yarn/*,  
$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*
```

mapreduce.app-submission.cross-platform

Handles various platform issues that arise if your Spectrum server runs on a Windows machine, and you install Cloudera on it. If your Spectrum server and Cloudera are running on different Operating Systems, then enter the value of this parameter as `true`. Else, mark it as `false`.

Note: Cloudera does not support Windows clients. Configuring this parameter is a workaround and not a solution to all resulting platform issues.

If you have checked the **Kerberos** checkbox above, then add the below

Stage or Activity using the HDFS Connection	Required Server Properties
	<p data-bbox="623 352 1110 384">Kerberos configuration properties additionally:</p> <p data-bbox="623 386 1055 420">hadoop.security.authentication</p> <p data-bbox="807 430 1333 499">The type of authentication security being used. Enter the value <code>kerberos</code>.</p> <p data-bbox="623 520 1068 554">yarn.resourcemanager.principal</p> <p data-bbox="807 564 1349 672">The Kerberos principal being used for the resource manager for your Hadoop YARN resource negotiator.</p> <p data-bbox="807 690 1357 724">For example, <code>yarn/_HOST@HADOOP.COM</code></p> <p data-bbox="623 745 1084 779">dfs.namenode.kerberos.principal</p> <p data-bbox="807 789 1357 896">The Kerberos principal being used for the namenode of your Hadoop Distributed File System (HDFS).</p> <p data-bbox="807 917 1357 951">For example, <code>hdfs/_HOST@HADOOP.COM</code></p> <p data-bbox="623 972 1068 1005">dfs.datanode.kerberos.principal</p> <p data-bbox="807 1016 1343 1123">The Kerberos principal being used for the datanode of your Hadoop Distributed File System (HDFS).</p> <p data-bbox="807 1144 1357 1178">For example, <code>hdfs/_HOST@HADOOP.COM</code></p>

Stage or Activity using the HDFS Connection

Required Server Properties

<ul style="list-style-type: none"> • Stage Read from File • Stage Write to File • Stage Read from Hive ORC File • Stage Write to Hive ORC File 	<p><i>Hadoop 1.x Parameters</i></p> <p>fs.default.name</p> <p>Specifies the node and port on which Hadoop runs.</p> <p>For example,</p> <pre>hdfs://152.144.226.224:9000</pre>
	<p><i>Hadoop 2.x Parameters</i></p> <p>fs.defaultFS</p> <p>Specifies the node and port on which Hadoop runs.</p> <p>For example,</p> <pre>hdfs://152.144.226.224:9000.</pre> <p>Note: For Spectrum versions 11.0 and earlier, the parameter name <code>fs.defaultfs</code> must be used. Note the case difference.</p> <p>For versions 11 SP1 and onwards, both the names <code>fs.defaultfs</code> and <code>fs.defaultFS</code> are valid. We recommend using the parameter name <code>fs.defaultFS</code></p> <p>Spectrum™ Technology Platform 11 SP1 onwards.</p>

15. To test the connection, click **Test**.

16. Click **Save**.

After you have defined a connection to an HDFS cluster, it becomes available in source and sink stages in Enterprise Designer, such as Read from File and Write to File. You can select the HDFS cluster when you click **Remote Machine** when defining a file in a source or sink stage.


Compression Support for Hadoop

Spectrum™ Technology Platform supports the compression formats `gzip (.gz)` and `bzip2 (.bz2)` on Hadoop. While using the **Read from File** and **Write to File** stages with an HDFS connection, include the extension corresponding to the required compression format (`.gz` or `.bz2`)

in the **File name** field. The file is decompressed or compressed based on the specified compression extension. Spectrum™ Technology Platform handles the compression and decompression of the files.

Connecting to a JDBC Database

In order for Spectrum™ Technology Platform to access data from a JDBC database, you must define a database using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to a JDBC database.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.


5. In the **Type** field, select type of database you want to connect to.

Spectrum™ Technology Platform Data Integration Module includes JDBC drivers for SQL Server, Oracle, and PostgreSQL databases. If you want to connect to a different database type, you must add the JDBC driver before defining a connection.

6. In the **URL** field, enter the JDBC connection URL. Your database administrator can provide this URL.

For example, to connect to a MySQL database named "SampleDatabase" hosted on a server named "MyServer" you would enter:

```
jdbc:mysql://MyServer/SampleDatabase
```

7. There may be additional fields you need to fill in depending on the JDBC driver. The fields represent properties in the connection string for the JDBC driver you selected in the **Type** field. See the JDBC driver provider's documentation or your database administrator for information about the specific connection properties and values required by the connection type.
8. Click **Save**.
9. Test the connection by checking the box next to the new connection and clicking the Test button .

Importing a JDBC Driver

Spectrum™ Technology Platform can access data from any database using a JDBC driver. Drivers for SQL, Oracle, and PostgreSQL are provided with Spectrum™ Technology Platform Data Integration Module, as well as drivers for other types of databases. If Spectrum™ Technology Platform Data

Integration Module does not come with a driver for the type of database you need, you can add a JDBC driver.

In this procedure you will import a JDBC driver by copying the driver files to the Spectrum™ Technology Platform server. When you complete this procedure the driver will be available to use when defining a JDBC database connection in Management Console.

Note: This procedure works for JDBC 4.x drivers. If the driver you want to add uses an older version of JDBC you must add the driver manually in Management Console. For more information, see [Manually Adding a JDBC Driver](#) on page 38

1. Put all the JDBC driver files for the database into a folder named:

Name.jdbc

Where *Name* is any name you want. The folder name must end with .jdbc.

2. Log in to the server running Spectrum™ Technology Platform.
3. Copy the folder containing the driver to this folder:

Spectrum Location\server\app\drivers

The driver is automatically imported.

4. To verify that the driver was successfully imported, log in to Management Console and go to **System > Drivers**. The driver should be listed.

If the driver is not listed, open the System Log in Management Console and look for errors related to deploying JDBC drivers.


Manually Adding a JDBC Driver

Spectrum™ Technology Platform can access data from any database using a JDBC driver. Drivers for SQL, Oracle, and PostgreSQL are provided with Spectrum™ Technology Platform Data Integration Module, as well as drivers for other types of databases. If Spectrum™ Technology Platform Data Integration Module does not come with a driver for the type of database you need, you can add a JDBC driver.

In this procedure you will add JDBC driver files to the server then manually define the connection string and connection properties. Before you begin, be sure that you understand the connection string format and properties required by the driver. You must define these accurately in order for the driver to function. You can typically find information about a driver's connection string and properties from the driver provider's website.

Note: We recommend that you use this procedure only when adding a JDBC driver that uses JDBC 1.x, 2.x, or 3.x. If the driver uses JDBC 4.x, we recommend that you use the import method to add the driver. For more information, see [Importing a JDBC Driver](#) on page 37.

1. Open Management Console.

2. Go to **System > Drivers**.
3. Click the Add button .
4. In the **Name** field, enter a name for the driver. The name can be anything you choose.
5. In the **JDBC driver class name** field, enter the Java class name of the driver. You can typically find the class name in your JDBC driver's documentation.

For example, to use the Microsoft JDBC driver, you might enter the following:

```
com.microsoft.sqlserver.jdbc.SQLServerDriver
```

6. In the **Connection string template** field, enter the JDBC connection URL to use to connect to the database, including any properties you want to set in the connection string. Different database vendors use different connection strings so check your database's documentation for more information about the connection string.

If the driver will be used by more than one database connection, consider using property tokens in the connection string instead of hard-coding property values that may be different for each connection. For example, if you want to have some connections use encryption and others not, you may want to define a property token for the encryption property.

To use a property token in the connection string, use this syntax:

```
${PropertyToken}
```

Any property tokens you include in the connection string template will be required fields when defining a database connection.

Note: Use the property token name `${password}` for the property that will contain the database password. By using this token name, the password will be masked in the field in Management Console and will be encrypted in the database.

For example, this connection string for SQL contains property tokens for host, port, instance, and encryption:

```
jdbc:sqlserver://${host}:${port};databaseName=${instance};encrypt=${encryption};
TrustServerCertificate=true
```

These tokens are required fields when defining a database connection that uses this driver:

Add Data Source

*Name

Connection


*Type

*Host

*Port

*Instance

*encryption

7. If there are properties that you want to make optional for database connections, define them in the **Connection Properties** section.
 - a) In the **Connection properties** section, click the Add button .
 - b) In the **Label** field, enter a user-friendly description of the property. The label you enter here is used as the field label in the connections window when creating a connection using this driver.
 - c) In the **Property token** field, enter the token for the optional property. See the database driver's documentation for the properties supported by the driver.


Note: Use the property token name `password` for the property that will contain the database password. By using this token name, the password will be masked in the field in Management Console and will be encrypted in the database.

For example, if you want to make encryption optional for database connections that use this driver, you could define the encryption property like this:

[Home](#) / [System: Drivers](#) / [Edit Driver](#)

Edit Deployed Driver

*Name

*JDBC driver class name *Connection string template 

Properties & Drivers

Connection properties 

Label	Property Token
<input type="checkbox"/> username	user
<input type="checkbox"/> password	password
<input type="checkbox"/> Use SSL	useSSL

When a database connection uses this driver, the encryption property would be displayed as an optional property in the database connection:

[Home](#) / [Data Sources](#) / [Add Data Source](#)

Add Data Source

*Name

Connection

*Type

*Host

*Instance

User Name

Password

 Use SSL

8. Log in to the server running Spectrum™ Technology Platform and place the database driver file in a folder on the server. The location does not matter.
9. In the **Driver files** section, click the Add button .
10. In the **File path** field, enter the path to the database driver file on the server.
11. Click **Save**.

Deleting an Imported JDBC Driver

JDBC drivers cannot be deleted using Management Console if the JDBC driver was imported to Spectrum™ Technology Platform rather than being added manually in Management Console. Instead, follow this procedure to delete the driver.

Important: Before deleting a driver, verify that there are no database connections using the driver.

1. Stop the Spectrum™ Technology Platform server.
2. Go to this folder:

Spectrum Location\server\app\drivers

3. In the `drivers` folder, delete folder containing the driver.
4. Start the Spectrum™ Technology Platform server.
5. To verify that the driver has been deleted, log in to Management Console, go to **System > Drivers**, and verify that the driver is no longer listed.

Supported Database Data Types

Spectrum™ Technology Platform supports these data types commonly used in databases:

bigdecimal	A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.
boolean	A logical type with two values: true and false.
date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.
Raw	An Oracle datatype for storing variable length binary data. Maximum size is 2000 bytes (the maximum length in Oracle 7 was 255 bytes).

Other database data types are automatically mapped to one of the supported data types as follows:

Database Data Type	Supported Data Type
Date/Time Types	
TIMESTAMP	datetime
String Types	
CHAR	string
CLOB	string
LONGVARCHAR	string
NCHAR	string
NVARCHAR	string
VARCHAR	string
Numeric Types	
BIGINT	long
DECIMAL	double
FLOAT	double
NUMERIC	bigdecimal
REAL	float
SMALLINT	integer
TINYINT	integer
Boolean Types	
BIT	boolean

Supported Database Data Types for the Location Intelligence Module

These database data types are automatically mapped to one of the supported data types for the Location Intelligence Module.

Database Data Type	Supported Data Type
SQL Server	
tinyint	SHORT_INTEGER
smallint	SHORT_INTEGER
int	INTEGER
bigint	LONG_INTEGER
float	DOUBLE
real	DOUBLE
decimal(10, 5)	DOUBLE
numeric(10, 5)	DOUBLE
date	DATE
time	TIME
datetime	DATE_TIME
smalldatetime	DATE_TIME
char(10)	STRING
varchar(10)	STRING
nchar(10)	STRING
nvarchar(10)G	STRING

Database Data Type	Supported Data Type
binary(10)	BINARY
varbinary(10)	BINARY
PostGIS	
smallint	SHORT_INTEGER
integer	INTEGER
bigint	LONG_INTEGER
numeric(10, 5)	DOUBLE
real	DOUBLE
double precision	DOUBLE
serial	INTEGER
bigserial	LONG_INTEGER
bytea	BINARY
date	DATE
time	TIME
timestamp	DATE_TIME
character(10)	STRING
character varying(10)	STRING
nchar(10)	STRING

Database Data Type	Supported Data Type
Oracle	
NUMBER	DOUBLE
CHAR(10)	STRING
VARCHAR(10)	STRING
VARCHAR2(10)	STRING
NCHAR(10)	STRING
NVARCHAR2(10)	STRING
DATE	DATE_TIME
TIMESTAMP	DATE_TIME
BLOB	BINARY
SAP HANA	
tinyint	SHORT_INTEGER
smallint	SHORT_INTEGER
integer	INTEGER
bigint	LONG_INTEGER
smalldecimal	DOUBLE
decimal(10, 5)	DOUBLE
real	DOUBLE
double	DOUBLE


Database Data Type	Supported Data Type
float(30)	DOUBLE
varchar(30)	STRING
nchar(10)	STRING
nvarchar(30)	STRING
alphanum(30)	STRING
date	DATE
time	TIME
seconddate	DATE_TIM
timestamp	DATE_TIM
varbinary(30)	BINARY

Limitations

- MongoDB/Cassandra connectors are not supported via PrestoDB in Metadata-Insights. There are separate connectors for MongoDB and Cassandra.
- Write to Any DB via Presto is not recommended by Presto DB and so is not supported by Presto JDBC connector.

Connecting to Knox

An Apache Knox Gateway allows you to access a Hadoop service through the Knox security layer. In order for Spectrum™ Technology Platform to access data in Hadoop via Knox, you must define a connection to Knox using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Hadoop via Knox.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.


5. In the **Type** field, choose *Gateway*.
6. In the **Gateway Type** field, choose **Knox**.
7. In the **Host** field, enter the hostname or IP address of the node in the HDFS cluster running the gateway.
8. In the **Port** field, enter the port number for the Knox gateway.
9. In the **User Name** field, enter the user name for the Knox gateway.
10. In the **Password** field, enter the password to authorize you access to the Knox gateway.
11. In the **Gateway Name** field, enter the name of the Knox gateway you wish to access.
12. In the **Cluster Name** field, enter the name of the Hadoop cluster to be accessed.
13. In the **Protocol** field, choose *webhdfs*.
14. In the **Service Name** field, enter the name of the Hadoop service to be accessed.
15. To test the connection, click **Test**.
16. Click **Save**.

After you have defined a Knox connection to an HDFS cluster, the connection can be used in Enterprise Designer, in the stages **Read from File** and **Write to File**. You can select the HDFS cluster when you click **Remote Machine** when defining a file in a source or sink stage.

Connecting to Marketo

In order for Spectrum™ Technology Platform to access data in Marketo, you must define a connection to Marketo using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Marketo.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Marketo**.
6. In the **Endpoint URL** field, enter the endpoint URL of your Marketo account.

To find your endpoint URL, log in to your Marketo account and go to **Admin > Integration > Web Services**. The endpoint URL is under the heading **REST API** and is in this format:

```
https://AccountID.mktorest.com/rest
```

Copy the portion of the URL before `/rest`. For example,
`https://AccountID.mktorest.com`.

7. Enter the client ID and secret key for your Marketo account.
 To find your client ID and secret key, log in to your Marketo account and go to **Admin > Integration > LaunchPoint > API Rest > View Details**. The pop-up window displays the details.
8. To test the connection, click **Test**.
9. Click **Save**.

Marketo Limitations

1. This query applies only to `List` and `Activity_type` entities. For others, provide the filter type.

```
Select * from Marketo_Table
```

2. Marketo does not support joins except between `Lead` and `Lead_List` entities. The join query between `Lead` and `Lead_List` for a `List_Id` is as follows:

```
Select Lead.* from Lead Inner Join Lead_List  
On Lead.ID = Lead_List.Lead_ID  
And Lead_List.List_ID = <List ID>
```


Connecting to Microsoft Dynamics 365

Connecting to Microsoft Dynamics 365 Online

In order for Spectrum™ Technology Platform to access data in Microsoft Dynamics 365 Online, you must define a connection to Microsoft Dynamics 365 Online using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Microsoft Dynamics 365 Online.

Note: This connection is for use in the Metadata Insights module.

Follow these steps to define a **Microsoft Dynamics 365 Online** connection.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .

- In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

- In the **Type** field, choose **Microsoft Dynamics 365**.
- In the **Development Type** field, select **Online**.
- In the **Username** field, enter your Microsoft Dynamics user name.
- In the **Password** field, enter your Microsoft Dynamics password.
- In the **Organization Name** field, enter your organization unique name, which identifies your CRM instance.

To find your organization unique name, log in to Microsoft Dynamics and go to **Settings > Customization > Customizations > Developer Resources**. Your organization unique name is displayed.

- In the **Region** field, select the geographical region of your Microsoft Dynamics account.
- To test the connection, click **Test**.
- Click **Save**.

Connecting to Microsoft Dynamics 365 On Premises

Currently, Spectrum supports Claims Based Authentications for Microsoft Dynamics 365 On Premises.

Prerequisites:

Import Certificate to the Keystore File: To import the Dynamics CRM Server certificates to the Spectrum Java distribution Keystore, do the following:

- Copy the server certificates to a local folder
- Browse the following path to the Spectrum JAVA distribution:
<SPECTRUM_HOME>\java\jre\lib\security
- Run the following command to import the certificates : <codeph>keytool -importcert -alias <certificate


alias name> -file " <certificate path>\<certificate name>" -keystore keystore.jks</codeph> in Windows and <codeph>keytool -import -alias <certificate alias name> -file "<certificate path>/<certificate name>" -keystore keystore.jks</codeph> in Unix

Defining Microsoft Dynamics 365 On Premises connection

To enable Spectrum™ Technology Platform to access data in Microsoft Dynamics 365 On Premise, define a connection to Microsoft Dynamics 365 OnPremise using Management Console. After defining the connection, you can create flows in Enterprise Designer that can read data from, and write data to, Microsoft Dynamics 365 On Premise.

Note: This connection is for use in the Metadata Insights module.

Do the following to define a **Microsoft Dynamics 365 On Premises** connection:

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. Click **Microsoft Dynamics 365** in the **Type**.
6. Click **On Premise** in the **Development Type**.
7. Enter your Microsoft Dynamics user name in the **Username**.
8. Enter your Microsoft Dynamics password in the **Password**.
9. Enter the name of the host in the **Host Name**.
10. Enter the name of the port In the **Port Name**.
11. Enter the URL of the STS in the **STS URL**.
12. Click **Test** to test the connection.
13. Click **Save**.


Limitations

Following are the limitations:

1. **Create/Update:** Create/Update can fail if a column in an Entity is mapped to multiple Reference Entities. For example, 'ParentCustomerId' in Customer can be associated to Account, Lead, and more. To resolve this, the data for this column needs to be in the following format: 'ReferenceEntityName:GUID' in place of 'GUID'

Connecting to a Model Store

Connect to a model store to use the data federated from various sources such as databases, file servers, and cloud services. Once a connection is defined, you can use the data in the logical and physical models of a model store in the **Read from DB** and **Write to DB** stages of Enterprise Designer.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Model Store**.
6. In the **Model store** field, enter the name of the model store that you are establishing connection with.

To find the names of the available model stores, open Metadata Insights, to Modeling, and click the **Model Store** tab.
7. To test the connection, click **Test**.
8. Click **Save**.

Connecting to NetSuite


In order for Spectrum™ Technology Platform to access data in NetSuite, you must define a connection to NetSuite using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, NetSuite. While reading from and writing to a NetSuite connection, both interactive and batch modes are supported.

Note: This connection is for use in the Metadata Insights module.

Spectrum™ Technology Platform supports these NetSuite entity types:

- Standard records
- Custom records
- Saved searches
- Joins between Standard records

To connect to NetSuite:

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **NetSuite**.
6. In the **Email** field, enter the e-mail linked to the NetSuite account to be used for the connection.
7. In the **Password** field, enter the password of the NetSuite account.
8. In the **Account** field, enter the user name for the NetSuite account.
9. In the **Role** field, select the appropriate role for this connection from the roles mapped to the particular NetSuite user account.

The **Role** field is optional. If you leave the **Role** field blank, the default role is used to log in through the connection.

Attention: Only standard roles are supported. Custom roles are not supported.

10. To test the connection, click **Test**.
11. Click **Save**.

Note: To `INSERT` a record using a NetSuite connection, use an `UPSERT` query with the primary key (`internalId`) blank.

NetSuite Limitations

1. When querying using joins, you must cite specific columns. For example, this query is not supported:

```
select * from CUSTOMER_M
```

2. Simultaneous connections to NetSuite are not supported because NetSuite allows only a single login per account.
3. You can only write Standard and Custom records.
4. For both `UPDATE` and `UPSERT` queries, an `UPSERT` operation is performed.
5. In the Write to DB stage, the maximum batch size permitted for an `insert` operation is 200 and the maximum batch size for an `update` operation is 100.
- 6.


Connecting to NoSQL

In order for Spectrum™ Technology Platform to access data in a NoSQL database, you must define a connection to the NoSQL database using Management Console.

The supported NoSQL database types are:

1. Couchbase
2. MongoDB

Once the desired NoSQL connection is defined, you can create flows in Enterprise Designer that can read data from and write data to this database.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, select any one of:


- Couchbase
 - MongoDB
6. Specify the **Host**, **Port**, **Database**, **Username** and **Password** of the specific NoSQL database you wish to access.
 7. Click **Test** to check that the connection to the database is successful.
 8. Click **OK**.

Connecting to Oracle Eloqua

For Spectrum™ Technology Platform to access data in Oracle Eloqua, you must define a connection to Oracle Eloqua using the Management Console.

. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Eloqua.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, click **Oracle Eloqua**.
6. Enter the **Site Name** which is same as Company Name.
7. Enter the user name in the **Username** field.
8. Enter the password in the **Password** field.
9. Click **Test** to test the connection..
10. Click **Save**.

Supporting Entities

Following table describes supporting entities and operations for Oracle Eloqua.

Table 1:

Entity Name	Create	Read	Update	Delete	Batch Support	Maximum Batch Size
Accounts	X	X	X	X	Insert/Update*	1000

Entity Name	Create	Read	Update	Delete	Batch Support	Maximum Batch Size
Account Groups		X				
Campaign		X				
Contacts	X	X	X	X	Insert/Update*	1000
Contact List	X	X	X	X		
Contact Segment	X	X	X	X		
Emails		X				
Email Folders		X				
Email Groups		X				
Microsites		X				
Users		X				
Visitors		X				
Activity						
Email Open		X				
EmailClickthrough		X				
Email Send		X				
Subscribe		X				
Unsubscribe		X				
Bounceback		X				

Entity Name	Create	Read	Update	Delete	Batch Support	Maximum Batch Size
WebVisit		X				
PageView		X				
FormSubmit		X				
Custom Entities						
ContactListMembers	X	X		X	Insert/Delete	1000
ContactSegmentMembers		X				

* Update operation works as Upsert.

Special Operations

1. Use the following join query to fetch contacts in a Contact List:

```
select * from Contacts inner join ContactListMembers on
Contacts.Eloqua_Contact_ID = ContactListMembers.Contact_Id where
ContactListMembers.ContactList_Id = '<id>'
```

Use the following join query to fetch contacts in a Contact Segment:

```
select * from Contacts inner join ContactSegmentMembers on
Contacts.Eloqua_Contact_ID = ContactSegmentMembers.Contact_Id where
ContactSegmentMembers.Contactlist_Id = '<id>'
```

2. Use the following statement to insert contacts in a Contact List:

```
insert into ContactListMembers (ContactList_ID,Contact_ID) values
('<contactlist_id>','<contact_id>')
```

- Use the following statement to delete contacts from a Contact List:

```
delete from ContactListMembers where ContactList_ID =
'<contactlist_id>' and Contact_ID = '<contact_id>'
```

Limitations

Note these limitations:

- Create/Update:**

- Insert/Upsert fails if Not Null columns are blank or do not exist
- Insert/Upsert fails if values of Unique columns are not unique for a particular batch
- In order to avoid a rollback exception, keep the value of **Batch count to commit** to 1

- Read:**

- For custom entities, Select operation is only applicable on joins with Contacts entity


- Filter:**

- Supported filters are =, !=, >, <, >=, <=
- There is no support for IN and NOT IN condition operators when providing more than one values
- There is no support for Joins between entities
- OR conditional operator is supported only for Accounts and Contacts entities
- AND conditional operator can be used only between two conditions
- = filter does not always work on fields having timestamp data type

Connecting to Salesforce

In order for Spectrum™ Technology Platform to access data in Salesforce, you must define a connection to Salesforce using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Salesforce.

Note: This connection is for use in the Metadata Insights module.

- Open Management Console.
- Go to **Resources > Data Sources**.
- Click the Add button .
- In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Salesforce**.
6. In the **Username** field, enter the email ID registered on the Salesforce data store.
7. In the **Password** field, enter a combination of the Salesforce portal password and the security token generated through the Salesforce portal.

For example, if your password is Sales@Test, and the security token provided to you by Salesforce is 56709367, then the Password to authenticate this Salesforce connection would be Sales@Test56709367.

8. To test the connection, click **Test**.
9. Click **Save**.

Note: Audit fields are enabled on all tables by default. The Salesforce audit fields are:

- created date
- last modified date
- created by
- last modified by

Attention: Physical model created in Spectrum™ Technology Platform version 10 and earlier using Salesforce connections need to be opened and saved again in order to enable audit fields on their tables.


Salesforce Limitation

The `Aggregate functions` are not supported while executing queries on Model Store.

Connecting to SAP NetWeaver

Creating a SAP NetWeaver connection in the Management Console using OData Services allows you to read, write and synchronize your CRM and ERP data. While reading from and writing to a SAP connection, both interactive and batch modes are supported.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **SAP**.

6. In the **Username** field, enter the username to access the SAP web service.
7. In the **Password** field, enter the password of the SAP web service.
8. In the OdataURL field, enter the address of the Odata web service to be used for this connection.
9. Click **Test**.
A message confirms the successful test of the connection.
10. Click **Save**.
A message confirms the successful creation of the connection.


Note: To perform fetch operations, an OData service must support the `$skip` and `$top` operations. If the service does not support these operations, the fetched records show inconsistencies in the model store preview.

SAP NetWeaver Limitations

For both `UPDATE` and `UPSERT` operations, an `UPDATE` operation is performed.

Connecting to SharePoint

In order for Spectrum™ Technology Platform to access data in SharePoint, you must define a connection to SharePoint using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, SharePoint.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Cloud**.
6. In the **Cloud service** field, choose **Sharepoint**.
7. In the **Version** field, select **v2010**. Spectrum™ Technology Platform currently supports SharePoint version 2010.
8. In the **Protocol** field, select the protocol required to connect SharePoint.
9. In the **Server address** field, enter the host name or IP address of the SharePoint server you want to connect to.
10. Enter the user name and password to use to authenticate to SharePoint.
11. In the **Project** field, enter the specific project whose SharePoint location you want to access.
12. To test the connection, click **Test**.
13. Click **Save**.

Example

For example, say you want to create a connection to this SharePoint URL:

```
https://sharepoint.example.com/sites/myportal
```


You would fill in the **Protocol**, **Server address**, and **Project** fields as follows:

- **Protocol:** https
- **Server address:** sharepoint.example.com
- **Project:** myportal

Connecting to Siebel

In order for Spectrum™ Technology Platform to access data in Siebel, you must define a connection to Siebel using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to Siebel.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **Siebel**.
6. In the **Host** field, enter the host name of the server where Siebel is installed.
7. In the **Port** field, enter the listening port number for the SCBroker component.

Note: Use 2321 for SCBPort.

8. In the **Server** field, enter the name of your Siebel Enterprise Server.
9. In the **Object Manager** field, enter the name of the Object Manager.
10. Enter the user name and password to use to authenticate to Siebel.
11. In the **Locale** field, select the type of Object Manager and Language Pack for the Siebel Business application.

Note: For locales other than **English**, you must install locale-specific JAR files. For more information, see [Required JAR Files](#).

12. To test the connection, click **Test**.
13. Click **Save**.

Within a physical model, Siebel business components are displayed in the format <Business Object>.<Business Component>.

Note: Business Components that are not a part of any Business Object are not displayed in physical model.

Siebel Limitations

1. Siebel Business Components fields with the same names but differing cases are handled by leaving the first occurrence unchanged, while appending all subsequent occurrences with `_PB_` and an increasing numeral value.

For example, fields like `DeDup Token`, `Dedup Token`, and `DEdup Token` in a Siebel business component are renamed to `DeDup Token`, `Dedup Token_PB_1`, and `DEdup Token_PB_2` respectively.

2. If a field name contains a dot (.) in a Siebel schema, the dots are removed from the field name for use in Spectrum™ Technology Platform.
3. While creating a physical model from a Siebel connection, links between any two Siebel Business Components are displayed only if the link name matches the Business Component name in the Siebel schema.


For example, in a Siebel schema, if a link is defined between the Business Components `Account` and `Contact`, while another Business Component `Contact Custom` uses the links of `Custom`, then in the physical model created for this Siebel connection, no link is displayed between `Account` and `Contact Custom`. Links are displayed only between `Account` and `Contact`, as the original link is between these two in the Siebel schema.

4. Fields of a Business Component which are marked as inactive in the Siebel schema are not displayed in physical model, logical model, and model store.

Connecting to Splunk

In order for Spectrum™ Technology Platform to access data in Splunk, you must define a connection to Splunk using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, Splunk.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .

- In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

- In the **Type** field, choose **Splunk**.
- In the **Username** field, enter the Splunk account user name to authenticate the Splunk instance.
- In the **Password** field, enter the password of the Splunk account.
- In the **Host** field, enter the address or host name of the server on which the Splunk data source is hosted.
- In the **Port** field, enter the port number of the Splunk data source.
- To test the connection, click **Test**.
- Click **Save**.

Splunk Limitations


This query is not supported:

```
select count(*) from SplunkTable
```

Connecting to SuccessFactors

In order for Spectrum™ Technology Platform to access data in SuccessFactors, you must define a connection to SuccessFactors using Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, SuccessFactors.

Note: This connection is for use in the Metadata Insights module.

- Open Management Console.
- Go to **Resources > Data Sources**.
- Click the Add button .
- In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

- In the **Type** field, choose **SuccessFactors**.
- In the **Company ID** field, enter your company ID, which identifies your company's unique instance in a specific SuccessFactors data center.
- In the **Service URL** field, enter the URL for the SuccessFactors server you want to connect to. This URL is specific to the global data center to which your company ID is mapped.
- Enter your user name and password for your SuccessFactors client instance.
- To test the connection, click **Test**.

10. Click **Save**.


SuccessFactors Limitations

1. Batch operations can only be performed using `upsert` queries. So, `insert` and `update` queries are also performed as `upsert` queries in batch operations.
2. The table/column properties, as displayed in the physical model schema of a SuccessFactors connection, might not function as expected during use of the corresponding operation. For example, a column, which has been marked `updatable`, may throw a system exception when you attempt to update that column.

Connecting to SugarCRM

For Spectrum™ Technology Platform to access data in SugarCRM, you must define a connection to SugarCRM using the Management Console. Once you do this, you can create flows in Enterprise Designer that can read data from, and write data to, SugarCRM. Both on-line and on-premise versions of SugarCRM are supported.

Note: This connection is for use in the Metadata Insights module.

1. Open Management Console.
2. Go to **Resources > Data Sources**.
3. Click the Add button .
4. In the **Name** field, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

5. In the **Type** field, choose **SugarCRM**.
6. Enter your SugarCRM user name and password.
7. In the **URL** field, enter the URL of the SugarCRM account to be used for this connection.
8. Enter the **Client Id** and **Client Secret** of your SugarCRM account.
9. To test the connection, click **Test**.
10. Click **Save**.

SugarCRM Limitations

1. For both `UPDATE` and `UPSERT` queries, an `UPSERT` operation is performed.
2. The **Nullable** and **Updatable** columns in the table properties, as displayed in the **physical model Schema** of the connection, may not represent the correct operation. For example, a column not marked as `Updatable` may throw system exception when you try to update it and conversely, a column marked as `Nullable` might not throw an exception in updating.

3. When querying using joins you need to use an alias.

Connecting to a Windows Mapped Drive

When Spectrum™ Technology Platform is running on a Windows server, it can access data on the server's mapped drives. Since the Spectrum™ Technology Platform server runs as a Windows service under a particular user account (often the Local System account) you need to define the mapped drive in the server's start-up process in order for it to be visible in Enterprise Designer and Management Console.

1. Stop the Spectrum™ Technology Platform server.
2. Under the folder where the Spectrum™ Technology Platform server is installed, go to `server\bin\wrapper`. For example, `C:\Program Files\Pitney Bowes\Spectrum\server\bin\wrapper`.
3. Open the file `wrapper.conf` in a text editor.

Important: In the following steps you will add new properties to this file. It is important that you follow these instructions precisely and only add and modify the properties described in the following steps. Do not modify any of the other properties in this file.

4. Add these lines:

```
wrapper.share.1.location
wrapper.share.1.target
wrapper.share.1.type
wrapper.share.1.account
wrapper.share.1.password
```

5. In the `wrapper.share.1.location` property, specify the location of the mapped drive in UNC format.

Note: Do not include a trailing backslash in the UNC.

For example,

```
wrapper.share.1.location=\\myserver\share
```

6. In the `wrapper.share.1.target` property, specify the drive letter to assign to this mapped drive.

For example,

```
wrapper.share.1.target=Y:
```

7. In the `type` property, specify `DISK`.

For example,

```
wrapper.share.1.type=DISK
```

8. If the share you are connecting to requires a user name and password, specify the user name in the `wrapper.share.1.account` property and specify the password in the `wrapper.share.1.password` property.

For example,

```
wrapper.share.1.account=domain\user123
wrapper.share.1.password=mypassword1
```


Note: If the Spectrum™ Technology Platform server service is running under the Local System user, you cannot specify a user name and password. If the share requires a user name and password you must modify the service to run under a different account.

Example

This example shows two mapped drives being defined in the `wrapper.conf` file.

```
wrapper.share.1.location=\\myserver\data
wrapper.share.1.target=Y:
wrapper.share.1.type=DISK
wrapper.share.1.account=sample\user
wrapper.share.1.password=samplepass
wrapper.share.2.location=\\myserver\moredata
wrapper.share.2.target=Z:
wrapper.share.2.type=DISK
wrapper.share.2.account=sample\user
wrapper.share.2.password=samplepass
```

Deleting a Connection

1. Open the Management Console.
2. Go to **Resources > Data Sources**.
3. Check the box next to the connection you want to delete, then click the Delete button .

3 - Populating the Data Warehouse

In this section

Preparing Your Data	68
Populating a Time Dimension Table	69
Populating a Dimension Table	70
Populating a Fact Table	72
Adding a Time Stamp to Records in a Data Warehouse	76

Preparing Your Data

Before populating your data warehouse, you should ensure that your data is of good quality, and that it has all the attributes necessary to provide meaningful insights to business users. One common approach is to use an operational data store (ODS) for this purpose. An ODS is a database where you can perform data quality operations that prepare your data for loading into the data warehouse. Spectrum™ Technology Platform has a variety of features that can be implemented in an ODS to improve the quality of your data and also augment your data with spatial, demographic, or other data. If you currently do not have the following features licensed, contact Pitney Bowes for more information.

Parsing, Name Standardization, and Name Validation

To perform the most accurate standardization you may need to break up strings of data into multiple fields. Spectrum™ Technology Platform provides advanced parsing features that enable you to parse personal names, company names, and many other terms and abbreviations. In addition, you can create your own list of custom terms to use as the basis of scan/extract operations. The Universal Name Module provides this functionality.

Deduplication and Consolidation

Identifying unique entities enables you to consolidate records, eliminate duplicates and develop "best-of-breed" records. A "best-of-breed" record is a composite record that is built using data from other records. The Advanced Matching Module and Data Normalization Module provide this functionality.

Address Validation

Address validation applies rules from the appropriate postal authority to put an address into a standard form and even validate that the address is a deliverable address. Address validation can help you qualify for postal discounts and can improve the deliverability of your mail. The Universal Addressing Module and the Address Now Module provide this functionality.

Geocoding

Geocoding is the process of taking an address and determining its geographic coordinates (latitude and longitude). Geocoding can be used for map generation, but that is only one application. The underlying location data can help drive business decisions. Reversing the process, you can enter a geocode (a point represented by a latitude and longitude coordinate) and receive address information about the geocode. The Enterprise Geocoding Module provides this functionality.

Location Intelligence

Location intelligence creates new information about your data by assessing, evaluating, analyzing and modeling geographic relationships. Using location intelligence processing you can verify locations

and transform information into valuable business intelligence. The Location Intelligence Module provides this functionality.

Tax Jurisdiction Assignment

Tax jurisdiction assignment takes an address and determines the tax jurisdictions that apply to the address's location. Assigning the most accurate tax jurisdictions can reduce financial risk and regulatory liability.

Spectrum™ Technology Platform software from Pitney Bowes integrates up-to-date jurisdictional boundaries with the exact street addresses of your customer records, enabling you to append the correct state, county, township, municipal, and special tax district information to your records. Some example uses of tax jurisdiction assignment are:

- Sales and use tax
- Personal property tax
- Insurance premium tax

The Enterprise Tax Module provides this functionality.

Populating a Time Dimension Table

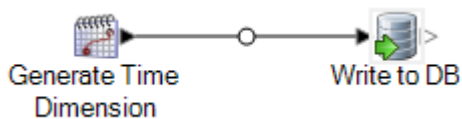
A time dimension table is a table in a database that makes it possible to analyze historic data without using complex SQL calculations. For example, you can analyze your data by workdays versus holidays, weekdays versus weekends, by fiscal periods or by special events.

The following procedure describes how to use Spectrum™ Technology Platform to populate a time dimension table in your data warehouse.

Note: Before beginning this procedure, you must have defined connections to the data warehouse in which you want to create a time dimension table. If you have not defined the necessary connection, see [Data Source connections](#) on page 14.

1. In Enterprise Designer, select **File > New > Dataflow > Job**.
2. Drag the Generate Time Dimension stage onto the canvas.
3. Drag a Write to DB stage onto the canvas and connect the Generate Time Dimension stage to it.

The dataflow should now look like this:



4. Double-click the Generate Time Dimension stage and configure it to produce the time dimensions you want. For more information, see [Generate Time Dimension](#) on page 107.

Note: The Julian day is usually used as a key value for a time dimension table if the grain is a day or more. If the grain is less than a day then you can generate a separate key by adding a Unique ID Generator stage to the dataflow. If you use the Julian day as the key, configure Generate Time Dimension to produce an integer column for Julian day values, and a column with the data type of date or datetime for date values.

5. Double-click the Write to DB stage on the canvas and configure it to point to the database and table where you want to create the time dimension table. For information on configuring Write to DB, see [Write to DB](#) on page 227.
6. To preview the time dimension values before writing them to the time dimension table:
 - a) Right-click the channel connecting the Generate Time Dimension stage and the Write to DB stage, and select **Add Inspection Point**.
 - b) Select **Run > Inspect Current Flow**.
The inspection pane appears at the bottom of the Enterprise Designer window and shows the data that will be written to the time dimension table. If necessary, you can make adjustments to the Generate Time Dimension stage and then re-run the inspection process to view the effect of your changes.
7. When you are satisfied with the dataflow, select **Run > Run Current Flow** to execute the dataflow and populate the time dimension table.

Populating a Dimension Table

Dimension tables are part of a star schema and contain detailed information for the columns in the fact table. Dimension tables have attributes and a single part primary key that joins the dimension table to the fact table. The single part primary key allows you to quickly browse a single dimension table. Browsing a dimension table can help determine the best way to query the fact table.

The following procedure describes how to use Spectrum™ Technology Platform to perform the initial population of a dimension table in your data warehouse.

Note: Before beginning this procedure, you must have defined connections to external resources you want to use as the source for the dimension table if you are using a database, file server, or web service as the source for the data. You must also define a connection to the data warehouse in which you want to create a dimension table. If you have not defined the necessary connections, see [Data Source connections](#) on page 14.

1. In your data warehouse, create the table that you want to use as a dimension table.
2. In Management Console, create connections to your data source and the data warehouse.
 - To connect to a database, see [Connecting to a JDBC Database](#) on page 37.
 - To connect to a file server, see [Connecting to an FTP Server](#) on page 26.

3. In Enterprise Designer, select **File > New > Dataflow > Job**.
4. Drag the source stage onto the canvas.
 - To use data from a database to populate the table, drag the **Read from DB** stage onto the canvas.
 - To use data from a flat file to populate the table, drag the **Read from File** stage onto the canvas.
 - To use data from a variable format file to populate the table, drag the **Read from Variable Format File** stage onto the canvas.
 - To use data from an XML file to populate the table, drag the **Read from XML** stage onto the canvas.
5. Double-click the source stage you just placed on the canvas and configure it to point to the source of the data you want to populate to the dimension table.
 - For information on configuring Read from DB, see [Read From DB](#) on page 128
 - For information on configuring Read from File, see [Read From File](#) on page 136
 - For information on configuring Read from Variable Format File, see [Read from Variable Format File](#) on page 183
 - For information on configuring Read from XML, see [Read From XML](#) on page 197
6. Drag a Unique ID Generator stage onto the canvas and connect the source stage to it. For example, if you are using Read from DB as the source stage, you would connect Read from DB to Unique ID Generator.
7. Double-click the Unique ID Generator stage on the canvas and configure it to create a surrogate key.

Note: Typically the key from the operational system is not used as the primary key for a dimension in the warehouse. This helps maintain historical consistency because a key value might change in the operational system.
8. Drag a Write to DB stage onto the canvas and connect Unique ID Generator to it.
9. Double-click the Write to DB stage on the canvas and configure it to point to the database and dimension table that you want to populate. For information on configuring Write to DB, see [Write to DB](#) on page 227.
10. Select **File > Save** and save the dataflow.
11. To run the dataflow now and populate the dimension table, select **Run > Run Current Flow**.

Populating a Fact Table

After you have populated the dimension tables in your data warehouse, you are ready to populate the fact table. You populate a fact table with numeric measurements from tables in the OLTP database.

Important: You must populate the dimension tables before populating the fact table.

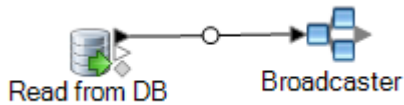
The following procedure describes how to use Spectrum™ Technology Platform to populate a fact table in your data warehouse. In this procedure, you will create a dataflow reads in source data from a table in your source database, replaces natural keys from the source tables with surrogate keys from the dimension tables, then loads the updated record containing the surrogate keys and the fact data from source tables into the fact table.

1. In Management Console, create connections to your data source and the data warehouse.
 - To connect to a database, see [Connecting to a JDBC Database](#) on page 37.
 - To connect to a file server, see [Connecting to an FTP Server](#) on page 26.
2. In Enterprise Designer, select **File > New > Dataflow > Job**.
3. Based on the source of the data you want to write to the fact table, drag the appropriate stage onto the canvas.
 - To use data from a database to populate the table, drag the **Read from DB** stage onto the canvas.
 - To use data from a flat file to populate the table, drag the **Read from File** stage onto the canvas.
 - To use data from a variable format file to populate the table, drag the **Read from Variable Format File** stage onto the canvas.
 - To use data from an XML file to populate the table, drag the **Read from XML** stage onto the canvas.
4. Double-click the source stage you just placed on the canvas and configure it to point to the source of the data you want to populate to the fact table.
 - For information on configuring Read from DB, see [Read From DB](#) on page 128
 - For information on configuring Read from File, see [Read From File](#) on page 136
 - For information on configuring Read from Variable Format File, see [Read from Variable Format File](#) on page 183
 - For information on configuring Read from XML, see [Read From XML](#) on page 197

Note: Typically, a dataflow that populates a fact table reads data from a database as opposed to a file. Because this is the most common scenario, the examples in the rest of this procedure use Read from DB.

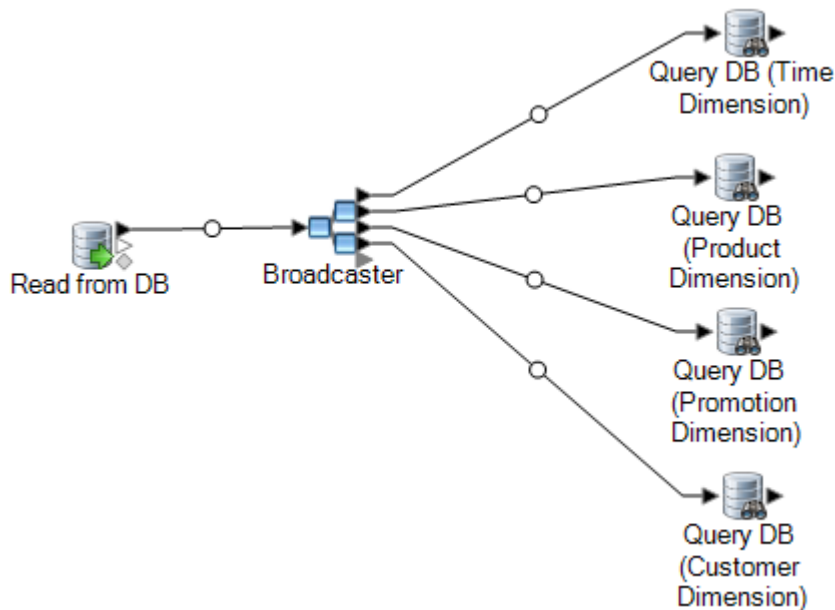
- Drag a Broadcaster stage onto the canvas and connect the source stage to it.

Your dataflow now looks like this:



- Drag one Query DB stage onto the canvas for each dimension table in your data warehouse and connect them to the Broadcaster stage.

For example, if you have four dimension tables in your data warehouse, drag four Query DB stages onto the canvas. Your dataflow would look like this:



The Query DB stages will be used to look up the surrogate key for each dimension using the natural key from the data source. The surrogate key will then replace the natural in each record being loaded into the fact table.

Tip: You can modify the name of the stage to make it easy to see which table each stage queries.

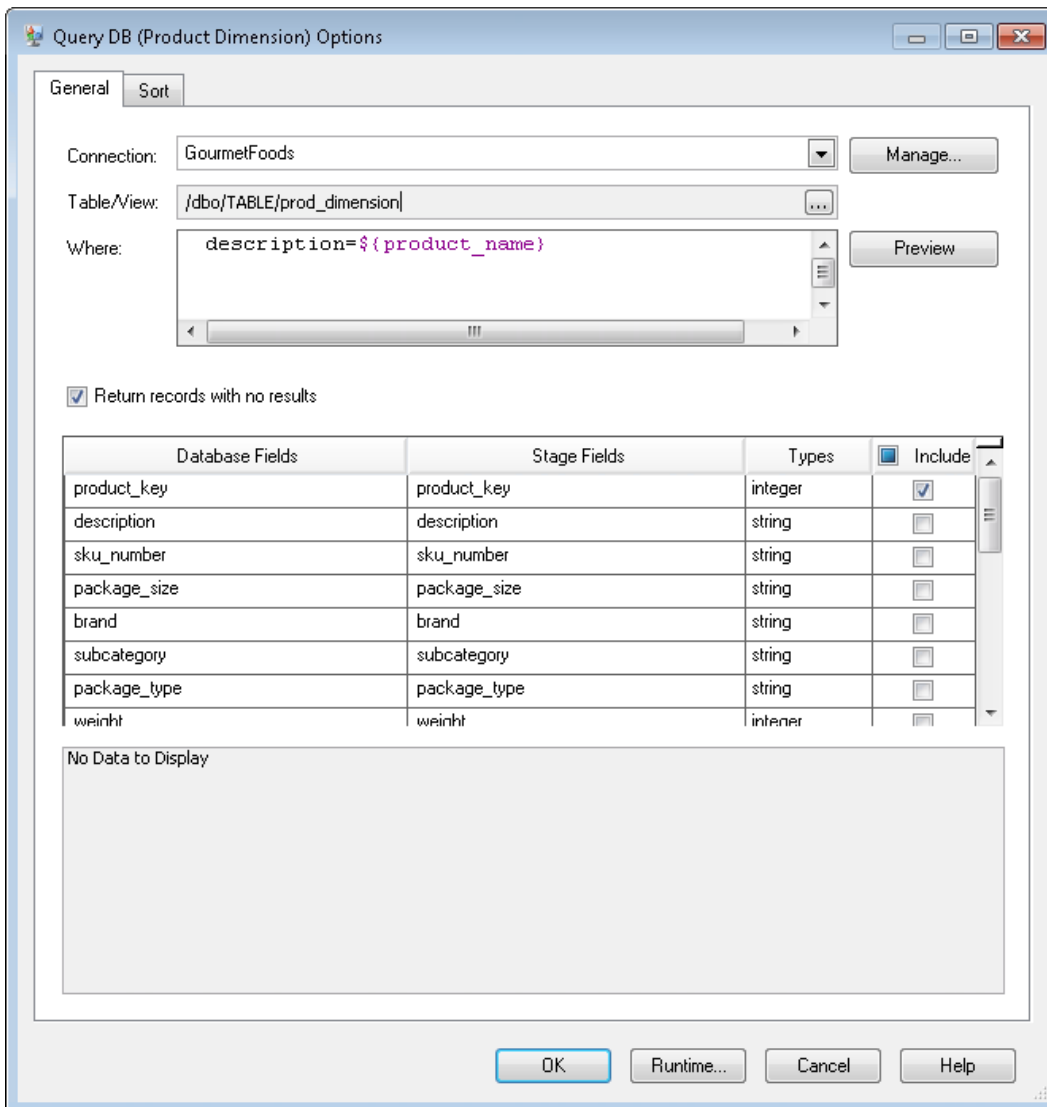
- Configure each Query DB stage so that it looks up the surrogate key for each natural key from the data source. To do this:
 - In the **Connection** field, specify the connection to the data warehouse.
 - In the **Table/View** field, select the dimension table that you want this stage to query.
 - In the **Where** field, write a `WHERE` statement that looks up the surrogate key based on the value in the appropriate dataflow field.

For example, this would look up the surrogate key for a product by finding the record in the dimension table whose value in the `description` column matches the value in the data source's `product_name` field.

```
description=${product_name}
```

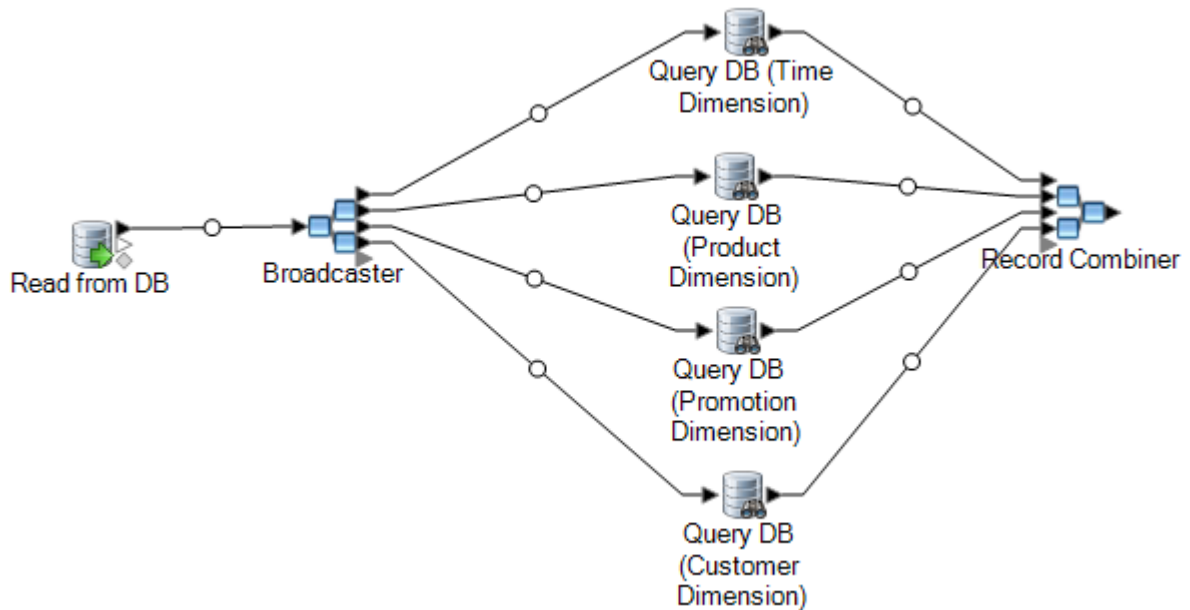
- d) In the **Include** column select the database column that contains the surrogate key.

For example, a Query DB stage that looks up the surrogate key for a product name would look like this:

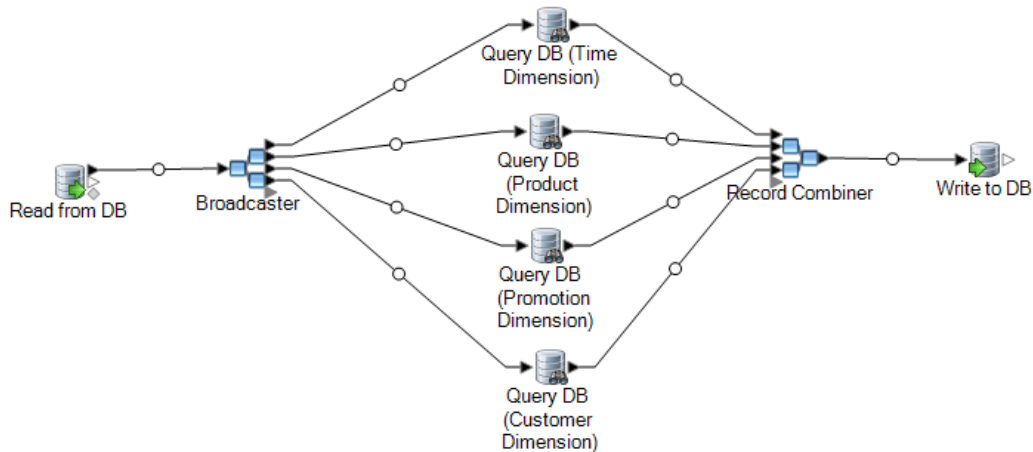


In this example, the query looks up the product key by finding the record in the `prod_dimension` table where the value in the `description` column matches the value in the dataflow field `product_name`. The stage returns the `product_key` field from the table and adds it to the dataflow, as indicated by the checked box in the **Include** column.

8. Drag a Record Combiner stage to the canvas and connect all the Query DB stages to it. Your dataflow should now look like this:



9. Drag a Write to DB stage onto the canvas and connect it to the Record Combiner stage. Your dataflow should now look like this:



10. Configure the Write to DB stage to write the records to the fact table. To do this:
 - a) In the **Connection** field, specify the connection to the data warehouse.
 - b) In the **Table/View** field, select the fact table that you want this stage to query. If the fact table does not already exist in the data warehouse, click **Create Table** to create the fact table in the data warehouse.
 - c) For each field that you want to write to the fact table, check the box in the **Include** column.
 - d) On the **Runtime** tab, notice that by default the **Insert** option is selected for the write mode. Typically fact table population is run in insert mode, so you can leave this option selected.

11. Save and run your dataflow.

Example of Replacing Source Data with Keys from the Dimension Table

Consider this record:

```
March 28 2013,Parsley Garlic Pasta,Mile High Gourmet
Market,78.35
```

In this example, there is a date field, followed by a product name (Parsley Garlic Pasta), a customer (Mile High Gourmet Market) and an amount (78.25). The data warehouse has dimension tables for the date, product name, and customer, so the natural keys in the record need to be replaced with the surrogate keys from the dimension tables. To accomplish this, the dataflow would have three Query DB stages, one that looks up the surrogate key for the date, one that looks up the surrogate key for the product name, and one that looks up the surrogate key for the customer.

Each Query DB would have a `WHERE` statement that looks up the surrogate key.

As a result of these lookups, the record might look like this when written to the fact table:

```
711,1,15,78.35
```

Notice that the natural keys for date, product name, and customer have been replaced with surrogate keys.

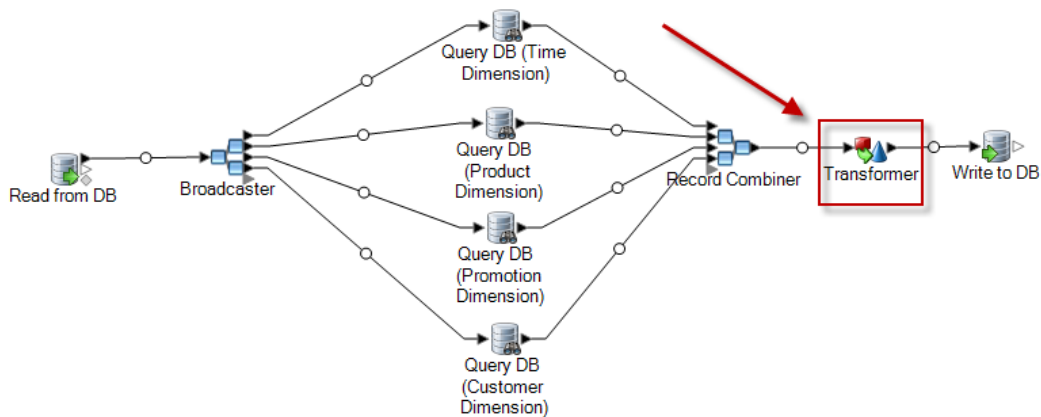
Adding a Time Stamp to Records in a Data Warehouse

A convenient way to assure data quality is to flag records in the data warehouse according to the date they were loaded. If the load process does not complete or you notice problems after data is already loaded, a time stamp column makes it easy to identify which records were affected. You can then delete all the records processed during a particular phase, return to the pre-load state and address any problems before attempting to load the data again. You can time stamp the load operation by adding an extra column, such as `load_date`, to your fact table using the SQL Command stage.

To have the dataflow add a timestamp when populating or updating a data warehouse:

1. In Enterprise Designer, open the dataflow that populates or updates the data warehouse:
2. Drag a Transformer stage onto the canvas and connect it to the dataflow just before the Write to DB stage.

For example:



3. Double-click the Transformer stage.
4. Click **Add**.
5. Under **General**, select **Custom**.
6. In the **Custom transform name** field, enter a name for this transform. The name can be anything you want. For example, Add Time Stamp.
7. In the **Custom script** field, enter the following:

```
data['<timestamp field>']=currentDateTime()
```

Where <timestamp field> is the name of the dataflow field that you want to contain the time stamp.

For example, if you want to put the time stamp in a dataflow field named `Timestamp` then your custom script would be:

```
data['Timestamp']=currentDateTime()
```

8. Click the **Add** button at the bottom of the window.
9. Click **Close**.
10. Click **OK** to close the **Transformer Options** window.

The dataflow now adds the current time to a field in each record, providing a time stamp in the data warehouse that shows when each record was loaded.

4 - Updating the Data Warehouse

In this section

Defining a Data Warehouse Update Schedule	79
Updating a Fact Table	80
Using a Global Cache for Queries	84
Using a Local Cache for Queries	85

Defining a Data Warehouse Update Schedule

You can schedule Spectrum™ Technology Platform dataflows to extract and transform data from the normalized structure in the data source into the star schema structure in the data warehouse. Scheduling dataflows is useful because most load operations require system resources that are unavailable during the business day.

When deciding on an update schedule, consider the following:

- Frequency
- Sequence
- Dependencies

Frequency

You should schedule dataflows to execute based on the grain of the most detailed fact table. For example:

- If the grain of the fact table is daily, schedule the fact table's population dataflow to run daily.
- If the grain is monthly, schedule the population dataflow to run monthly, not sooner, because users work only with data from completed past months.

Most population dataflows process large amounts of data, so schedule population dataflows to execute when use of the Spectrum™ Technology Platform server, the source and data warehouse databases, and the network is minimal.

Populate all dimension and fact tables during the initial load. After the initial load, refresh tables based on what was added or changed. Generally, fact tables are refreshed more frequently than dimension tables because:

- Dimension tables are usually static unless an attribute in the source is changed or added.
- Fact table data in a decision support database is typically historical and requires regular additions and updates to remain current. The initial load and most incremental loads affect fact tables.

Sequence

There are dependencies among data in the data warehouse databases, so determine the sequence in which to run population dataflows before setting the execution schedule.

Populate dimension tables before fact tables because every dimension record and key must exist before a related fact table can be populated. This restriction is a function of the primary-foreign key relationship between dimension and fact tables in a star schema.

Refresh base-level tables before populating aggregate tables in your decision support database. This sequence ensures that base-level and aggregate tables remain synchronized.

The correct order to run population dataflows is:

1. Base-level dimension table Plans
2. Base-level fact table Plans
3. Aggregate dimension table Plans
4. Aggregate fact table Plans

Dependencies

You can create dataflow dependencies if several population dataflows need to run in a specific order, or if the amount of time to run dataflows is unpredictable. A dataflow is run only if certain requirements are met such as, the previous dataflow has completed, or the previous dataflow has failed.

To create dataflow dependencies, create a process flow in Enterprise Designer. For more information about process flows, see the *Dataflow Designer's Guide*.

Updating a Fact Table

This procedure describes how to create a dataflow that reads data from a source database or file and uses that data to update a fact table in your data warehouse.

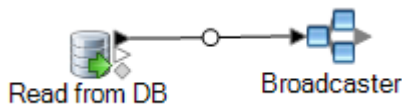
1. In Enterprise Designer, select **File > New > Dataflow > Job**.
2. Based on the source of the data you want to write to the fact table, drag the appropriate stage onto the canvas.
 - To use data from a database to populate the table, drag the **Read from DB** stage onto the canvas.
 - To use data from a flat file to populate the table, drag the **Read from File** stage onto the canvas.
 - To use data from a variable format file to populate the table, drag the **Read from Variable Format File** stage onto the canvas.
 - To use data from an XML file to populate the table, drag the **Read from XML** stage onto the canvas.

Note: If you will be reading data from a file and not a database, make sure that the file contains only the new records that you want to add to the fact table and not records that already exist in the fact table. If you will be reading data from a database, you will define a query to filter records later in this procedure.

3. Double-click the source stage you just placed on the canvas and configure it to point to the source of the data you want to populate to the fact table.
 - For information on configuring Read from DB, see [Read From DB](#) on page 128
 - For information on configuring Read from File, see [Read From File](#) on page 136

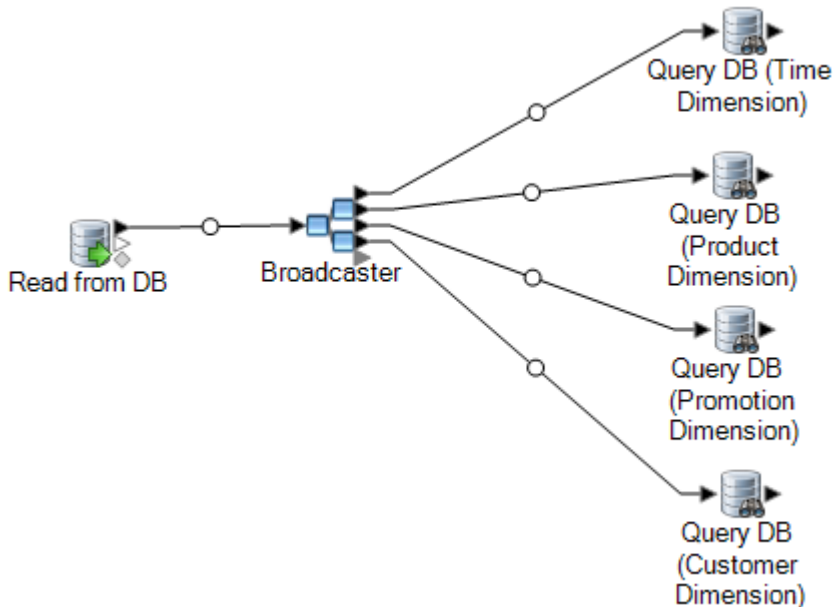
- For information on configuring Read from Variable Format File, see [Read from Variable Format File](#) on page 183
 - For information on configuring Read from XML, see [Read From XML](#) on page 197
4. If you are reading data from a database, filter the records so that only the records that are new will be added to the fact table. You can do this by defining the SQL SELECT statement to only read in records that were modified since the last time the fact table was updated.
 5. Drag a Broadcaster stage onto the canvas and connect the source stage to it.

Your dataflow now looks like this:



6. Drag one Query DB stage onto the canvas for each dimension table in your data warehouse and connect them to the Broadcaster stage.

For example, if you have four dimension tables in your data warehouse, drag four Query DB stages onto the canvas. Your dataflow would look like this:



The Query DB stages will be used to look up the surrogate key for each dimension using the natural key from the data source. The surrogate key will then replace the natural in each record being loaded into the fact table.

Tip: You can modify the name of the stage to make it easy to see which table each stage queries.

7. Configure each Query DB stage so that it looks up the surrogate key for each natural key from the data source. To do this:
 - a) In the **Connection** field, specify the connection to the data warehouse.

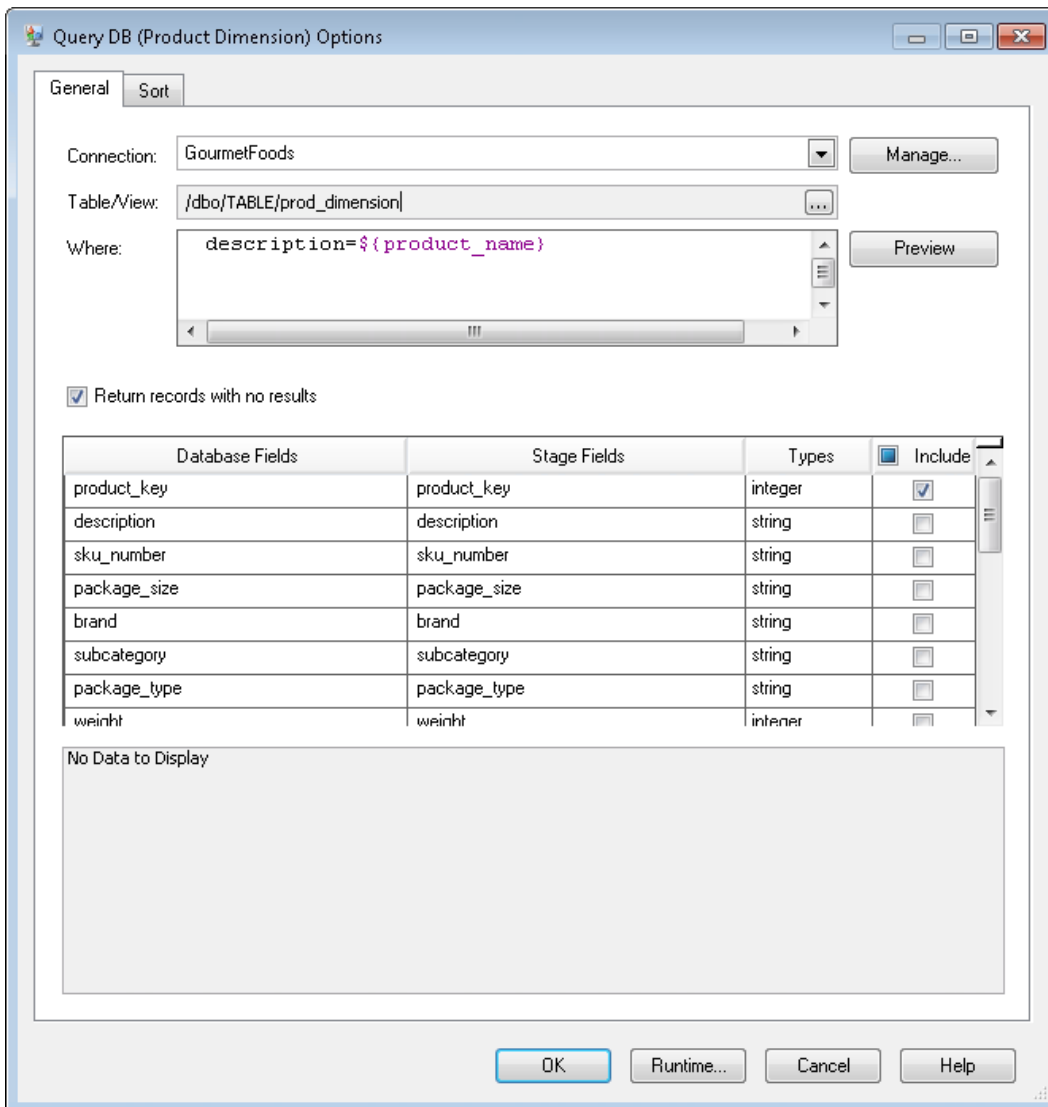
- b) In the **Table/View** field, select the dimension table that you want this stage to query.
- c) In the **Where** field, write a `WHERE` statement that looks up the surrogate key based on the value in the appropriate dataflow field.

For example, this would look up the surrogate key for a product by finding the record in the dimension table whose value in the `description` column matches the value in the data source's `product_name` field.

```
description=${product_name}
```

- d) In the **Include** column select the database column that contains the surrogate key.

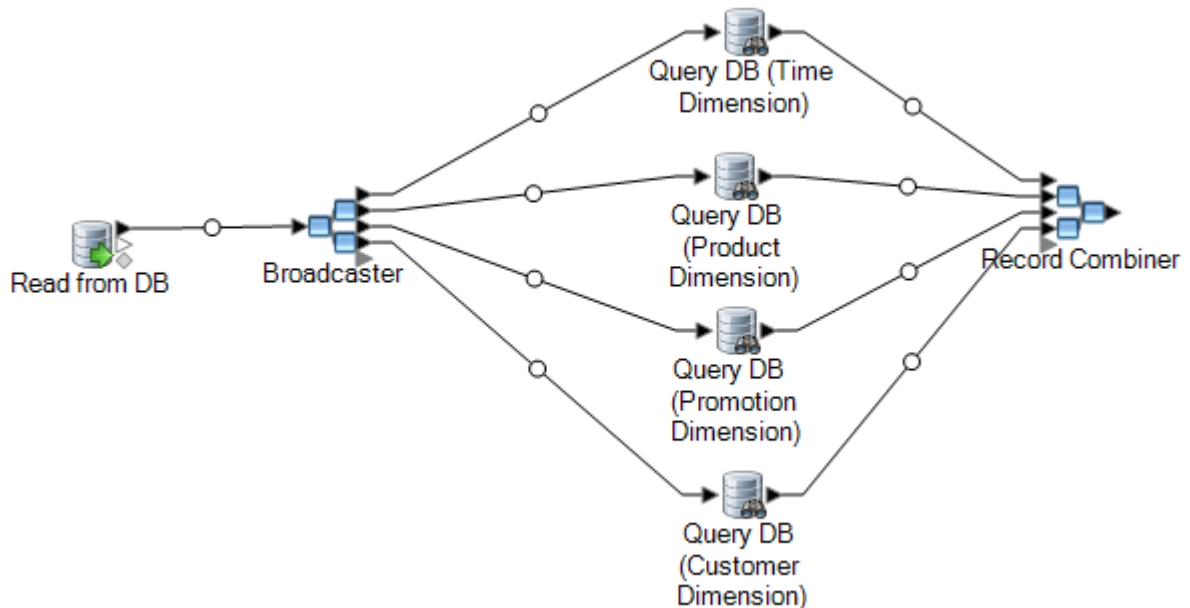
For example, a Query DB stage that looks up the surrogate key for a product name would look like this:



In this example, the query looks up the product key by finding the record in the `prod_dimension` table where the value in the `description` column matches the value in the dataflow field `product_name`. The stage returns the `product_key` field from the table and adds it to the dataflow, as indicated by the checked box in the **Include** column.

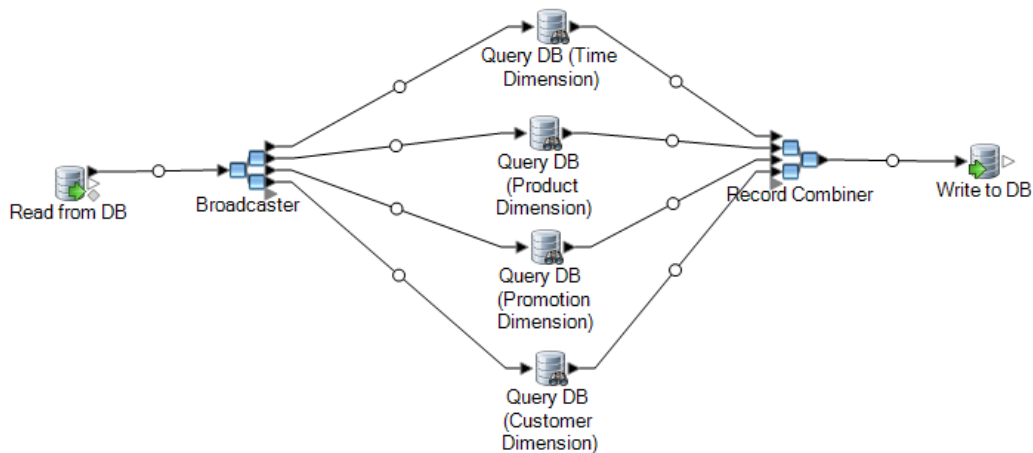
8. Drag a Record Combiner stage to the canvas and connect all the Query DB stages to it.

Your dataflow should now look like this:



9. Drag a Write to DB stage onto the canvas and connect it to the Record Combiner stage.

Your dataflow should now look like this:



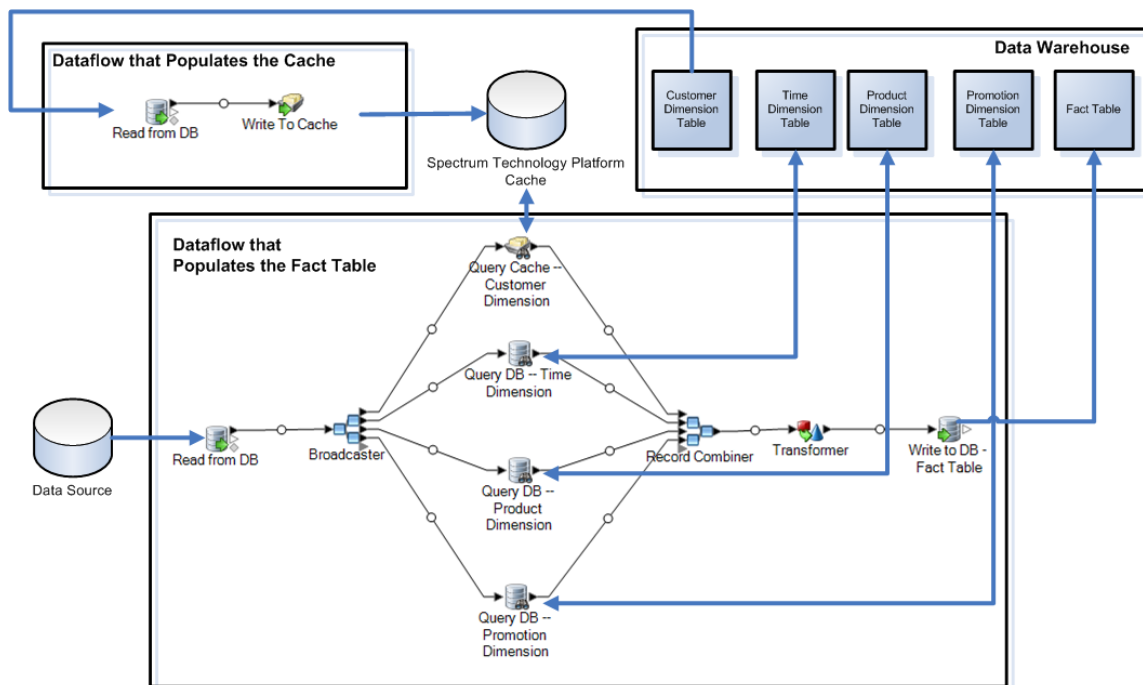
10. Configure the Write to DB stage to write the records to the fact table. To do this:
 - a) In the **Connection** field, specify the connection to the data warehouse.

- b) In the **Table/View** field, select the fact table that you want this stage to query. If the fact table does not already exist in the data warehouse, click **Create Table** to create the fact table in the data warehouse.
- c) For each field that you want to write to the fact table, check the box in the **Include** column.
- d) On the **Runtime** tab, notice that by default the **Insert** option is selected for the write mode. Typically fact table population is run in insert mode, so you can leave this option selected.

Using a Global Cache for Queries

If you have a large dimension table you can load the dimension table's data into a cache, and use the cache to look up surrogate keys. Using a cache improves performance compared to performing lookups directly to the dimension table with Query DB.

To use a cache you must create two dataflows: one to populate the cache with data from the dimension table, and another that uses the cache when updating the fact table. The following diagram illustrates how the two dataflows work together:



1. Create a dataflow that populates the cache with dimension table data from the large dimension table.

This dataflow should consist of two stages:

- A Read from DB stage that reads data from the dimension table that you want to load into the cache.

- A Write to Cache stage that populates the cache with the dimension table data.
2. Run this dataflow to populate the cache.
 3. In the dataflow that populates the fact table, add a Query Cache.
 4. In the Query Cache stage, configure the stage to query the cache created by the Write to Cache stage.
 5. Run this dataflow to populate the fact table.


If you want to make sure that the cache is populated with the latest data from the dimension table each time you update your fact table, you can create a process flow that first runs the job to populate the dimension table, then runs the job to update the fact table. This allows you to execute the process flow in order to run both dataflows in succession. For more information about process flows, see the *Dataflow Designer's Guide*.

Deleting a Cache

A global cache is used by the Query Cache stage for data lookups. If the cache is no longer needed, you can delete it by following this procedure. Deleting a cache frees up the memory.

Note: **Deleting a cache** is only used to delete a global cache from Management Console.

Warning: Before deleting a cache, ensure it is not being used by any dataflows containing a Query Cache stage. If you delete a cache used by a dataflow, the dataflow fails.

1. Open Management Console.
2. Expand **Resources > Cache Management**.
The grid displays the existing global caches.
3. Select the cache you wish to delete.
4. Click .

Using a Local Cache for Queries

If you have a large dimension table you can load the dimension table's data into a cache, and use the cache to look up surrogate keys. Using a cache improves performance compared to performing lookups directly to the dimension table with Query DB.

A local cache is a temporary cache which is only used during the execution of the Query Cache stage. It looks up data in the cache based on key fields and lookup conditions and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow. Use a

local cache instead of global cache if it is only going to be used in one dataflow or if the lookup table changes frequently.

To use a local cache for queries:

1. In Enterprise Designer, open the dataflow where you want to perform a query using a cache.
2. Drag a Query Cache stage onto the canvas and connect it to the dataflow.
3. Double-click the Query Cache stage.
4. Select **Local Cache**.
5. Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click **Manage**.

If you are adding or modifying a database connection, complete these fields:

Connection name	Enter a name for the connection. The name can be anything you choose.
Database driver	Select the appropriate database type.
Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.

6. Specify the table or view in the database that you want to query.
7. Select a key under the option **Key field**.
8. Select an input field under the option **Input field**. The **Input field** contains the fields coming from the previous stage. If the value in this field matches the key in the **Key field** in the database, then the query returns the data from that record in the database.
9. Click **OK**.
10. Run the dataflow.

5 - Stages Reference

In this section

Call Stored Procedure	88
DB Change Data Reader	90
DB Loader	93
Field Parser	102
Field Combiner	105
Field Selector	106
Generate Time Dimension	107
Query Cache	114
Query DB	122
Query NoSQL DB	125
Read From DB	128
Read From File	136
Read from Hadoop Sequence File	153
Read From Hive File	157
Read from HL7 File	160
Read from NoSQL DB	172
Read from SAP	176
Read from Spreadsheet	181
Read from Variable Format File	183
Read From XML	197
SQL Command	204
Transposer	213
Unique ID Generator	217
Write to Cache	225
Write to DB	227
Write to File	232
Write to Hadoop Sequence File	249
Write to Hive File	253
Write to NoSQL DB	259
Write to Spreadsheet	263
Write to Variable Format File	266
Write to XML	276
Date and Number Patterns	283

Call Stored Procedure

Call Stored Procedure is a source stage that executes a stored procedure in a database, and returns the results of the stored procedure call as input for the dataflow. Use Call Stored Procedure when you want to get data from a database using a database's stored procedure rather than a query to a table or view.

Note: If you want to read data into a dataflow directly from a table or view, use the Read from DB stage.

You may want to use Call Stored Procedure to read data into a dataflow if you have business logic embedded in the stored procedure and you want to use that logic in your Spectrum™ Technology Platform environment. For example, many operational systems do not use referential integrity checks in the database for large constantly-updated tables because of the reduction in performance that such checks would cause. So to maintain referential integrity, you could create stored procedures and use them for all updates to the system.

Stored procedures can also be used to simplify management of the Spectrum™ Technology Platform environment. For example, if you have hundreds of ETL processes that all read the same data, you may want to put the query into a stored procedure so it is in one place. This makes maintenance easier since you only need to modify the one stored procedure instead of hundreds of different processes.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Schema	Specifies the schema that contains the stored procedure you want to call.

Option Name	Description
Procedure	Specifies the stored procedure you want to call.
Stored Procedure Parameters	<p>This table specifies the values for the stored procedure parameters.</p> <p>Parameters This column shows the parameters defined in the stored procedure.</p> <p>Stage Fields For OUT, INOUT, and RETURN parameters, this column shows the dataflow field name that will contain the data returned by the parameter. Initially, the field name is the same as the parameter name. You can modify the stage field name by clicking the field name and typing a new name for parameters. This column is not used for IN parameters.</p> <p>Direction One of the following:</p> <ul style="list-style-type: none"> IN The parameter is an input parameter. The value you specify for this parameter is passed to the stored procedure as input. OUT The parameter is an output parameter. The stored procedure returns data to the stage in this parameter. INOUT The parameter can be used as both an input parameter to pass a value to the stored procedure, and as an output parameter to receive data returned by the stored procedure. RETURN The parameter contains a return code from the stored procedure. <p>Types This column displays the data type of the parameter value. If the data type is not supported by Spectrum™ Technology Platform, the type will be "Unsupported" and the stored procedure will not execute successfully.</p> <p>Value In this column, enter the value you want to set for the parameter. This column is disabled for OUT parameters.</p>

Option Name	Description
Result Set Fields	<p>This table specifies which dataflow fields to use for the data returned by the stored procedure.</p> <p>Database Tables This column shows the tables from which the stored procedure returned data.</p> <p>Database Fields This column shows the field from which the stored procedure returned data.</p> <p>Stage Fields This column shows the dataflow field name that will contain the data from the database field.</p> <p>Types This column shows the data type of the field. If the data type is not supported by Spectrum™ Technology Platform, the type will be "Unsupported".</p> <p>Include Check the box in this column to include the field in the dataflow. If the box is not checked, the field will not be used in the dataflow.</p>
Get Fields	Click this button to populate the Result Set Fields table with the result set schema returned by the stored procedure. This will execute the stored procedure and get the result set schema.
Add	Click this button to add a result set field manually.
Remove	Click this button to remove a result set field from the list of available fields.

DB Change Data Reader

The **DB Change Data Reader** stage allows you to select the columns to be included in the current jobflow, where the columns have the Change Data Capture feature enabled on them.

In the stage, you can create a Change Data Capture (CDC) Resource, which is the required data source table. The columns of the CDC resource on which the Change Data Capture feature is enabled are reflected using checkboxes.

Change Data Capture

The Change Data Capture feature enables capture of all changes made in a column. For each selected column, all inserts, updates, and deletions are captured.

Supported Databases

Currently, Spectrum™ Technology Platform supports the Change Data Capture (CDC) feature for MS SQL and Oracle databases only.

MS SQL For MS SQL data sources, the Change Data Capture feature can be enabled or disabled for columns of tables from the backend. Refer to [here](#) for the necessary steps.

Note: The Change Data Capture feature of Spectrum™ is not supported in the Express edition of SQL Server.

Oracle For Oracle data sources, Spectrum™ tracks the data changes in the table columns using the LogMiner utility of Oracle. CDC cannot be enabled or disabled on columns of tables in an Oracle data source through Spectrum™ Technology Platform.

The data changes resulting from `insert`, `update`, and `delete` queries are tracked and captured from an entered start date and time up to the current date and time. This entered start time is applicable during the first query execution with CDC switched on.

In subsequent executions on the same Oracle connection, data changes are captured from the time of the last execution till the current time.

Note: This is an incremental process. In the first capture, the changes are captured from the entered start date and time to the current date and time. In subsequent captures, the changes are captured from after the end date and time of the previous capture to the current date and time.

Adding a CDC Resource

Note: To use the Change Data Capture feature of Spectrum™, ensure the SQL Server Agent is running on the MS SQL server.

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Click **Add**.
3. Enter a **Name** for the CDC Resource.
4. In the **Connection** field, select the SQL database connection you want to use. To make a new database connection, click **Manage**. For more information on creating database connections, see [Database Connection Manager](#).
5. In the **Table/View** field, specify the table whose columns are to be included in the jobflow. Click the browse button ([...]) to navigate to the table or view that you want to use. The grid below displays all the columns of the selected table, along with the datatypes of each column.
6. If you select an Oracle connection in the **Connection** field, the **Start date** field becomes available.

The field is filled with the default value of the current date with the time 12:00 AM. You can enter a start date and time of your choice.

The end date and time is taken as the current date and time.

Attention: You should have *execution* rights on the Oracle LogMiner utility to be able to use the CDC feature on an Oracle connection. For more information, see [Oracle LogMiner Configurations](#) on page 291.

Note: The **Start date** field is not available on selecting an MS SQL connection in the **Connection** field.

7. Click **OK**.

The created CDC Resource table is now ready for use in the **DB Change Data Reader** stage, whose columns can be included or excluded from the jobflow.

The stage displays the table columns on which the CDC feature has been enabled.

Editing a CDC Resource

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Select the CDC Resource you need to modify.
3. Click **Edit**.
4. Modify the details of the added CDC Resource as required.
5. Click **OK**.

Deleting a CDC Resource

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Select the CDC Resource you need to delete.
3. Click **Delete**.

Selecting Change Data Reader Options

The **DB Change Data Reader Options** reflects the table columns of the selected CDC Resource for which the CDC feature is enabled.

You can select which columns to include or exclude in the current jobflow.

1. In a job, add the **DB Change Data Reader** stage.
2. Open the **DB Change Data Reader Options** by double-clicking the stage icon.
3. Select the desired CDC Resource from the **Select a resource..** dropdown.

You can add or modify a CDC Resource by clicking **Manage**.

The grid below displays all the table columns with their datatypes. It also displays whether a particular column is included in the job flow, and whether the column is selected for the Change Data Capture feature.

4. Using the checkboxes under the **Include** column of the grid, select the table columns to be included in the job flow.
5. The checkboxes under the **CDC Enabled** column of the grid reflect the table columns on which the CDC feature is enabled.

The read-only **CDC Enabled** checkboxes are checked for columns on which the CDC feature is enabled.

Note: For MS SQL data sources, refer [here](#) for the steps to enable or disable the Change Data Capture feature on particular table columns from the backend.

6. Click **OK**.

The data of the table columns, which have been selected for Change Data Capture, is captured and saved.

DB Loader

The **DB Loader** stage allows you to access and load data from/to databases configured in the Spectrum™ Data Integration Module. This stage provides an interface to a high-speed data loading

utility. Currently, the Spectrum™ Data Integration platform supports **Oracle Loader**, **DB2 Loader**, **PostgreSQL Loader**, and **Teradata Loader**.

Oracle Loader

The Oracle Loader allows you to load data to any Oracle database configured in the Spectrum™ Data Integration platform.

Note: Oracle client must be installed with administrator setup before using the Oracle Loader.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	<p>After selecting a connection, specify the table or view to write to. Click the browse button ([...]) to navigate to the table or view that you want to use, or click Create Table to create a new table in the database.</p> <p>Note: If you are writing to a SQL database, you cannot write to views that reference more than one table. This is due to a limitation in SQL Server.</p>
Listener	<p>Specify a variable name that contains the address and connection details required to establish a connection to the Oracle database. For example, "XE". This variable is present in <code>tnsnames.ora</code>, a file that contains client side network configuration parameters.</p>
Stage fields	<p>This column lists the field names used in the dataflow. You cannot modify these field names.</p>
Types	<p>This column lists the data type of each field.</p>

Runtime Tab

Option Name	Description
Load method	<p>Specifies how you want to write data into the Oracle database.</p> <p>Append Adds the data into the target table without erasing the table's existing data.</p> <p>Insert Loads the data into the database. The table must be empty before it is loaded. It doesn't work on multiple runtime instances.</p> <p>Truncate and Insert Deletes existing rows if any, then loads the input data into the table. This does not work on multiple runtime instances.</p>
Use direct path loader	Select this to load data directly into the Oracle database bypassing much of the data processing that normally takes place.
Unrecoverable	This check box is enabled when you select Use direct path loader . Select this if you do not want to write Redo logs in the database. For more information about Redo logs, see http://docs.oracle.com/cd/B28359_01/server.111/b28310/onlineredo001.htm#ADMIN11302
Log file folder	Specifies the path to the folder. Click the ellipses button (...) to browse to the folder you want. The log file contains a record of this stage's activities during a load session.
Bad file folder	Specifies the path to the folder. Click the ellipses button (...) to browse to the folder you want. The bad file contains a list of records that the stage fails to load into the database.
Maximum errors allowed	Specify the maximum number of errors to allow before halting the load operation. To halt the load operation on the first error, set value to 0. A maximum of 32767 errors are allowed.

Note: You can achieve significant performance improvements by using multiple runtime instances of this operation. To do this, click the **Runtime** button and enter the required value in the **Runtime instances** field.

DB2 Loader

The DB2 Loader allows you to load data to any DB2 database configured in the Spectrum™ Data Integration platform.

Note: DB2 runtime client must be installed with administrator setup before using the DB2 Loader.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	<p>After selecting a connection, specify the table or view to write to. Click the browse button ([...]) to go to the table or view that you want to use, or click Create Table to create a new table in the database.</p>
Database/Alias	<p>This is a variable that catalogues the DB2 server and database.</p> <p>To catalog the DB2 server Use the DB2 command line processor on spectrum server machine and enter the command:</p> <pre>CATALOG TCPIP NODE <nodename> REMOTE <hostname> SERVER <port></pre> <p>where:</p> <p>nodename: name of connection</p> <p>hostname: TCP/IP name of the DB2 server machine</p> <p>port: server port</p> <p>To catalog the database Use the command:</p> <pre>CATALOG DATABASE <databasename> AS <local_database_alias> AT NODE <nodename></pre> <p>where:</p> <p>databasename: Name of the database on the DB2 server</p> <p>local_database_alias: Local Name given to the database while connecting from the server machine</p> <p>nodename: Name used in the previous CATALOG TCP/IP command</p>
Stage fields	<p>This column lists the field names used in the dataflow. You cannot modify these field names.</p>

Option Name	Description
Types	This column lists the data type of each field.

Runtime Tab

Option Name	Description
Load method	<p>Indicates the mode of writing data into a DB2 table.</p> <p>Insert Inserts the loaded data into the table, while the existing table data remains unchanged.</p> <p>Replace Inserts the loaded data into the table after deleting all existing data from it.</p> <p> The table schema and index definitions remain unchanged.</p> <p>Restart Restarts the data load, in case the previous load attempt was interrupted.</p>
Non-recoverable	<p>Indicates if this load transaction is non-recoverable.</p> <p>If you select this option, the load transaction is marked as non-recoverable. Table spaces are not put into the Backup Pending state after the load, nor is a copy of the loaded data made during the load. Hence, a non-recoverable transaction cannot be recovered in the event of a data load failure, even if a <code>rollforward</code> is attempted later.</p> <p>If you select this option, you cannot recover from the transaction even if you use the DB2 <code>rollforward</code> utility because the utility skips such a non-recoverable transaction, and the table is marked as "invalid". In addition, subsequent transactions against the table are also ignored by <code>rollforward</code>.</p> <p>To restore a table that contains non-recoverable transactions, you must use either a tablespace-level backup or a full backup taken at a commit point following the non-recoverable load.</p> <p>Note: Do not select this option if the data contains Datalink columns that have the File Link Control attribute present in them.</p>
CPU	The number of parallel threads that the load utility can generate and sustain for loading, parsing and formatting the records, while building table objects in each database partition.
Disk	The number of parallel threads that the load utility can generate and sustain for writing data to table space containers.

Option Name	Description
Indexing Mode	<p>Indicates the mode of handling of indexes by the load utility.</p> <p>Autoselect The load utility decides whether to apply Rebuild or Incremental mode, based on the amount of data and the depth of the index tree.</p> <p>Rebuild All indexes are rebuilt.</p> <p>Incremental New data is added to the existing indexes.</p> <p>This mode can be applied only if the index object is valid and accessible at the start of a load operation.</p> <p>Note: Incremental indexing is not supported if ALL of these conditions hold true:</p> <ol style="list-style-type: none"> 1. The Load Copy option is specified (<code>logretain</code> or <code>userexit</code> is enabled). 2. The table resides in a DMS table space. 3. The index object resides in a table space that is shared by other table objects belonging to the table being loaded. <p>To bypass this limitation, place indexes in separate table spaces.</p> <p>Deferred The load utility does not attempt creating an index. Existing indexes are marked for refresh.</p> <p>Note: Index construction requires more time in Deferred mode than in Rebuild mode. Hence, while performing multiple load operations, allow the last load operation to rebuild all indexes instead of rebuilding indexes at the first access by a non-load operation.</p> <p>Note: This mode is supported only for tables with non-unique indexes.</p>
Fast Parse	<p>Indicates whether syntactical validation on column values must be left out, thus enhancing performance.</p> <p>If checked, any syntactical errors in the data are ignored in favor of optimized performance.</p> <p>For example, if a String value <code>12wxvg56</code> is encountered in a field mapped to an integer column in an ASCII file, the load utility should normally generate a syntax error. But if Fast Parse is selected, the syntax error is ignored, and a random number is loaded into the integer field.</p> <p>Note: Ensure you use this option only with correct and clean data.</p>
Schema Name	The schema in which the exception tables are stored.
Table Name	The exception table into which those rows are copied in which some error is encountered while loading.

Option Name	Description
Log file folder	<p>The path of the directory in which the log files are to be stored.</p> <p>A log file contains a list of the database load transactions run by a DB Loader stage in one load session.</p> <p>Click the ellipses button (...) to specify the desired directory for log files.</p>
Bad file folder	<p>The path of the directory on the DB2 server in which the bad files are to be stored.</p> <p>A bad file contains a list of the records that a DB Loader stage fails to load into the database.</p> <p>Click the ellipses button (...) to specify the desired directory for bad files.</p>
Maximum errors allowed	<p>The maximum number of errors allowed before a load operation is aborted.</p> <p>To abort a load operation as soon as the first error is encountered, set the value of this field to 0.</p> <p>Note: A maximum of 32767 errors are allowed.</p>
Parallelism	<p>A DB2 database can be divided into multiple partitions by cloning the environment onto different physical nodes.</p> <p>Separate database requests for data fetch and update are automatically divided amongst the different partitions and run in parallel for optimized performance.</p>
Exception Handling	<p>A DB2 database allows you to record the errors and exceptions encountered while running queries and procedures, and also handle them appropriately.</p> <p>For this, a DB2 database provides specific exception tables and schema which store the source as well as the log traces of each database exception.</p>

PostgreSQL Loader

The PostgreSQL Loader allows you to load data to any PostgreSQL database configured in the Spectrum™ Data Integration platform.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Click the browse button ([...]) to navigate to the table or view you want to use.
Database fields	This column lists the field name in the database. You cannot modify these field names.
Stage fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Types	This column lists the data type of each field.

Runtime Tab

Option Name	Description
Load method	<p>Specifies the method of writing data to the PostgreSQL database tables.</p> <ul style="list-style-type: none"> • Select Insert to write data to an empty table or to append it to an existing data table. • Select Truncate and Insert to truncate the data before loading it to the database table.

Note: You can achieve significant performance improvements by using multiple runtime instances of this operation. To do this, click the **Runtime** button and enter the required value in the **Runtime instances** field.

Teradata Loader

The Teradata Loader allows you to load data to any Teradata database configured in the Spectrum™ Data Integration platform.

Note: The loader is supported only on Windows systems.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Click the browse button ([...]) to navigate to the table or view you want to use.
Database fields	This column lists the field names in the database. You cannot modify these field names.
Stage fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Types	This column lists the data type of each field.

Runtime Tab

Option Name	Description
Load method	<p>Specifies the method of writing data to the Teradata database tables.</p> <ul style="list-style-type: none"> • Select Insert to write data to an empty table • Select Append to write data to an existing data table. • Select Truncate and Insert to truncate the data before writing it to the database table.
Log file folder	Select the folder location for saving the log files of the upload process.
Generate bad file	Mark this check-box to generate a log of the records that failed to get uploaded.
Bad file folder	Select the folder location for saving the log of failed records. The log gives details, such as error codes and error field names of the failed records.

Option Name	Description
Maximum errors allowed	Specify the limit of permitted errors for the upload session. If the number of errors exceeds this value, the upload process pauses.

Note: Multiple runtime instances are not supported for Teradata Loader.

Field Parser

The **Field Parser** stage extracts fields from XML and delimited data in the specified input column. To configure the **Field Parser** options, perform the following tasks.

1. From the **Source** field select the column that has the XML or delimited data to be parsed.

Note: The drop-down displays all the string input columns.

2. Select the XML or Delimited **Format** based on the type of data you want to parse, and accordingly, select the options described below.

Field Parser Options for XML Data

Option Name	Description
Server name	Indicates whether the file selected for inferring the schema is located on the computer running the Enterprise Designer or on the server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.
Schema file	<p>Specifies the path to an XSD schema file. Click the ellipses button (...) to navigate to the file location. The schema file can reside on the server or your local system.</p> <p>Alternatively, you can also specify an XML file instead of an XSD file. If you specify an XML file the schema will be inferred based on the structure of the XML file. Using an XML file instead of an XSD file has the following limitations:</p> <ul style="list-style-type: none"> • The XML file cannot be larger than 1 MB. If the XML file is more than 1 MB in size, try removing some of the data while maintaining the structure of the XML. • The data file will not be validated against the inferred schema. <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>

Option Name	Description
Output Fields	<p>This section displays details of the selected schema. It includes the root element followed by the child elements along with their attributes.</p> <p>By default all the nodes of the schema remain selected. However, you can clear the check-box of the nodes that you do not want to be passed to the next stage.</p> <ul style="list-style-type: none"> • Search node: Type the name of the node to which you want to navigate in the schema tree. The typed node gets highlighted in the preview pane below the field. • XPath: Click anywhere in this field to view the XML path (XPath) of the elements and attributes of the highlighted node in schema tree. To see all the previous XPaths viewed by you, click the down arrow at the right end of the field. <p>Note: XPath is a language for finding information in an XML document. For further details on this, see www.w3schools.com/xpath/</p>

Field Parser Options for Delimited Data

Option Name	Description
Field separator	<p>From the dropdown list, select the field separator used in the delimited column to be parsed.</p> <p>If the delimited column uses a different character as a field separator, click the ellipses button to select another character as field separator.</p>
Text qualifier	<p>From the dropdown list, select the text qualifier used in the delimited column to be parsed.</p> <p>Note: Text qualifiers are the character used to surround text values in a delimited data.</p> <p>If the delimited column uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Output type	<p>Select if you want the parsed output in the form of a List (hierarchical display of values) or Fields.</p> <p>Note: For list as the output type, you can add only one output field, whereas the Fields option allows you to add multiple fields in which you can get the values segregated during parsing.</p>

Option Name	Description
Output Fields	<p>This section allows you to add/modify the various fields in which you want details of the delimited column to be segregated. You can also delete any of the added output fields.</p> <p>To add a new field for displaying the parsed output, click the Add button, and perform these steps in the Field Setting pop-up that is displayed:</p> <ol style="list-style-type: none"> 1. Enter the Name of the field. 2. From the Type drop-down, select the data type for the field being added. Based on the selected type, few more fields can be defined. For example, in case of date, you can define its format as <code>M/d/yy</code>, <code>MMM d.yyyy</code>, or <code>MMMM d.yyyy</code>. For details on the data types and defining its details, see Defining Fields In a Delimited Input File on page 141. <p style="text-align: center;">Note: If you select <code>String</code> as the data type, any type of delimited data will be parsed. However, you can also use the specific type, based on the data you want to parse in the field.</p> 3. In the Position field, enter the position of the data type (in the input file) that is to be parsed to this field. For example, in the following file snippet, if you want to parse the date time values to the field being added, enter the Position as 3. <pre data-bbox="532 940 1424 1157">true;"02/02/2022";"10/2/92 5:05 AM";598985994665542.25634;1;"Arjun";74785.155;5:05PM,1,Deepak,65152 false;"15/03/1923";"3/23/90 11:55 AM";3425699466554.2563;2;"sharma";5.1;5:45AM,2,Arjun,365273</pre> 4. Click Add Field and Close. <p>The added field and its details are displayed in the box.</p> <p style="text-align: center;">Note: If you want to have any excess space characters removed from the beginning and end of a field's value string, select the Trim check box.</p> <p>Modify: Click this button to modify details of any of the added output fields.</p> <p>Remove: Click this button to delete any of the added output fields.</p>

Runtime: Use this button to specify multiple runtime instances of parser. This results in significant performance improvement.

OK: Click this button to save all the details entered in this stage.

Cancel: Click this button to cancel all the updates you made.

Help: Click this button to refer to the help file for this stage.

Field Combiner

The **Field Combiner** stage combines fields coming from the previous stage in a dataflow to create an XML string.

Field Combiner Options

Option Name	Description
Output column name	Specify the name of the column that will have the combined fields as XML string.
Output Fields	<p>This section helps you select the fields to be combined and perform various actions on those fields.</p> <p>Note: The Apply Namespace dropdown list and Element, Attribute, and Remove buttons get enabled when you select one or more elements/attributes in the schema pane.</p> <ul style="list-style-type: none"> • Quick Add: Click this button to open the Add/Remove Fields pop-up window. This window displays a list of all the fields coming from the previous stage. Select the fields you want to combine. <ul style="list-style-type: none"> Note: The selected fields are displayed in the schema pane below the Search node field. • Search node: Type the name of the node to which you want to navigate in the schema pane. The typed node gets highlighted and its XML path is displayed in the XPath field below the pane. • Element: Click this button to convert an attribute to an element in the XML string. • Attribute: Click this button to convert an element to an attribute in the XML string. • Apply Namespace: If you want to specify an XML namespace for an element or an attribute, select it here. <ul style="list-style-type: none"> Note: You can create namespaces by clicking the Namespace button above. For details on this, see Defining Namespaces. • Remove: Click this button to remove the elements/attributes that you do not want in the XML string.
XPath	<p>Click anywhere in this field to view the XML path (XPath) of the node highlighted in the schema pane. To see all the previous XPaths viewed by you, click the down arrow at the right end of the field.</p> <p>Note: XPath is a language for finding information in an XML document. For further details on this, see www.w3schools.com/xpath/</p>

- **Runtime:** Use this button to specify multiple runtime instances of Field Combiner. This results in significant performance improvement.
- **OK:** Click this button to save all the details entered in this stage.
- **Cancel:** Click this button to cancel all the updates you made.
- **Help:** Click this button to refer to the help file for this stage.

Defining Namespace

Namespaces allow you to have duplicate element and attribute names in your output by assigning each element or attribute to an XML namespace.

To define one or more namespaces:

1. On the **Field Combiner Options** screen, click the **Namespace** button. The **Namespace Details** pop-up window is displayed.
2. In the **Prefix** column, enter the prefix to be associated with an element or attribute.
3. In the **Namespace** column, specify the URL of the namespace.
4. Repeat to define as many namespaces as you want to use.

Field Selector

Field Selector allows you to choose which fields to pass to the next stage in the dataflow. You can use Field Selector to remove unwanted fields from a dataflow. For example, if you have created a new field by combining the data from two fields, and you no longer need the two source fields, you can use Field Selector to retain only the new field and remove the two source fields from the dataflow.

Options

Option	Description
Select the fields to send to the next stage	Check the box next to each field that you want to send to the next stage in the dataflow. To prevent a field from being sent on to the next stage, clear the check box.
Select all	Check this box to select all the fields in the dataflow. Clear this box to deselect all fields.

Generate Time Dimension

Generate Time Dimension creates date records, one for each day of the date range you specify. You can then write these records to a time dimension table in a database using the Write to DB stage. The time dimension table can then be used to perform accurate calculations based on a time period. For example, sales by quarter, budget spend by quarter, and revenue by day are all analyses that require a time dimension. Time dimension tables also enable you to account for fiscal years or non-standard quarters in the analysis.

Example Use of a Time Dimension Table

Time dimension tables are necessary for accurate time-based calculations because you sometimes cannot easily extract the necessary date data from the records. For example, the following records are in a sales database. Note that there are time gaps between records. For example, there is no record for the day 1/4/2012.

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/5/2012	Red Shirt	\$5.00
1/7/2012	Red Shirt	\$15.00

If you query these records and calculate the average sales per day, the answer would be \$10.00 ($\$30 / 3$ records). However, this is incorrect because the three records actually span a period of five days. If you have a time dimension table with a record for each day, you could join that table with the above table to get this:

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/4/2012		
1/5/2012	Red Shirt	\$5.00
1/6/2012		

Date	Product	Amount
1/7/2012	Red Shirt	\$15.00

Calculating the average sales per day using these records, you would get the correct answer: \$6.00 (\$30 / 5 days).

In addition, you could account for arbitrary time attributes such as holidays, weekends, and quarters in your calculation. For example, if 1/6/2012 happened to be a holiday and you were only interested in average sales per workday then the answer would be \$7.50.

Options

Generate Time Dimension has the following options.

Option	Description
Start date	The first day of the date range for which to generate time dimension records.
End date	Select this option if you want to specify a specific end date for the time dimension. One record will be generated for each day between the start date and the date you specify here.
Duration	Select this option if you want the time dimension to span a certain number of days, months, or years. One record will be generated for each day of the duration. For example, if you specify one week, seven records will be generated, starting with the day you specified in the Start date field.

Option	Description
Time Attribute	

Option	Description
	<p>Specifies the type of time information that you want to include in the time dimension. Each attribute will be a field in each day's record. One of the following:</p> <p>Date The day's date in the format <Date> <Month> <Year>, with the name of the month in the language of the server's locale setting. For example, if the server is running in an English locale, the date would read like this: 30 October 2012.</p> <p>Day of Month A number representing the day of the month. For example, 10 means the day is the 10th day of the month.</p> <p>Day of Year A number representing the day of the year. For example, 304 means that the day is the 304th day of the year.</p> <p>Is Leap Year A boolean value that indicates whether the day is in a leap year, in which case the value would be <code>true</code>, or not, in which case the value would be <code>false</code>.</p> <p>Is Weekday A boolean value that indicates whether the day is a weekday (Monday through Friday), in which case the value would be <code>true</code>, or not, in which case the value would be <code>false</code>.</p> <p>Is Weekend A boolean value that indicates whether the day is a weekend day (Saturday or Sunday), in which case the value would be <code>true</code>, or not, in which case the value would be <code>false</code>.</p> <p>Julian Day A unique number for the day. The Julian day system counts January 1, 4713 BC as day 1 and numbers all subsequent days sequentially since then. For example, 2456231 means that the day is the 2,456,231st day since January 1, 4713 BC.</p> <p>Julian Week A unique number for the day's week. The Julian system counts the first week of 4713 BC as week 1 and numbers all subsequent weeks sequentially since then. For example, 350890 means that the day is in the 350,890th week since the first week of 4713 BC.</p> <p>Julian Year A unique number for the year. The Julian system counts 4713 BC as year 1 and numbers all subsequent years sequentially since then. For example, 6725 means that the day is in the year 2012 AD.</p> <p>Month Name The name of the day's month in English.</p> <p>Month Number The number of the month of the year. For example, 3 means that the day is in the third month of the year.</p> <p>Quarter A number representing the quarter of the year. For example, 1 means that the day is in the first quarter of the year.</p> <p>Week of Year A number representing the week of the year. For example, 43 means that the day is in the 43rd week of the year.</p> <p>Weekday Name The name of the day in English. For example, Monday.</p> <p>Weekday A number representing the day of the week, with day 1 being</p>

Option	Description
	<p>Number Monday. So for example Tuesday is day 2, Wednesday is day 3, and so on.</p> <p>Year The day's year according to the Gregorian calendar. For example, 2012 means that the day is in the year 2012.</p>
Field	The name of the field that will be created in the dataflow to contain the time attribute. A default field name is provided but you can modify the field name.
Type	The data type of this field. Generate Time Dimension automatically chooses the appropriate data type for each time attribute.
Calendar	Specifies whether to use a custom calendar when calculating the time attribute or the default Gregorian calendar. To define a custom calendar, click Calendar .

Creating a Calendar

A calendar defines important characteristics of the year in a way that is appropriate to the type of analysis you want to perform. By default, the calendar is the standard Gregorian calendar. However, the standard Gregorian calendar is not always appropriate. For example, if your company's financial year begins on June 1 instead of January 1, you could define the first month of the year as June, and the "month of year" time attribute would have June as month 1, July as month 2, and so on, as opposed to January as month 1, February as month 2, and so on.

- Do one of the following:
 - If you are configuring the Generate Time Dimension stage in Enterprise Designer, click **Calendars**.
 - In Management Console, go to **Resources > Calendars**.
- Click **Add**.
- In the **Calendar name** field, give the calendar a name that is meaningful to you.
- In the **Start month** field, select the first month of the year.
For example, if you select July, then July is month 1, August is month 2, and so on.

Note: Changing the start month affects how the values in the following fields are calculated: DayOfYear, MonthNumber, Quarter, and WeekOfYear.

- Click **Save**.

Output

Generate Time Dimension produces the following output fields.

Table 2: Generate Time Dimension Output

Field Name	Description
Date	The day's date in the format <Date> <Month> <Year>, with the name of the month in the language of the server's locale setting. For example, if the server is running in an English locale, the date would read like this: 30 October 2012.
DayOfMonth	A number representing the day of the month. For example, 10 means the day is the 10 th day of the month.
DayOfYear	A number representing the day of the year. For example, 304 means that the day is the 304 th day of the year.
IsLeapYear	A boolean value that indicates whether the day is in a leap year, in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
IsWeekday	A boolean value that indicates whether the day is a weekday (Monday through Friday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
IsWeekend	A boolean value that indicates whether the day is a weekend day (Saturday or Sunday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
JulianDay	A unique number for the day. The Julian day system counts January 1, 4713 BC as day 1 and numbers all subsequent days sequentially since then. For example, 2456231 means that the day is the 2,456,231 st day since January 1, 4713 BC.

Field Name	Description
JulianWeek	A unique number for the day's week. The Julian system counts the first week of 4713 BC as week 1 and numbers all subsequent weeks sequentially since then. For example, 350890 means that the day is in the 350,890 th week since the first week of 4713 BC.
JulianYear	A unique number for the year. The Julian system counts 4713 BC as year 1 and numbers all subsequent years sequentially since then. For example, 6725 means that the day is in the year 2012 AD.
MonthName	The name of the day's month in English.
MonthNumber	The number of the month of the year. For example, 3 means that the day is in the third month of the year.
Quarter	A number representing the quarter of the year. For example, 1 means that the day is in the first quarter of the year.
WeekOfYear	A number representing the week of the year. For example, 43 means that the day is in the 43 rd week of the year.
WeekdayName	The name of the day in English. For example, Monday.
WeekdayNumber	A number representing the day of the week, with day 1 being Monday. So for example Tuesday is day 2, Wednesday is day 3, and so on.
Year	The day's year according to the Gregorian calendar. For example, 2012 means that the day is in the year 2012.

Query Cache

Query Cache looks up data in a cache based on values in one or more dataflow fields and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow. Looking up data in a cache can improve performance compared to looking up data in a database.

There are two kinds of caches: global caches and local caches.

Global Cache Options

A global cache is system-wide, shared cache that will reside in memory. Choose a global cache if you want the cache to be available to multiple dataflows or when data does not change often or remains relatively static and when storage is not limited. A global cache is static as you can write to it only once. The cache can not be updated once it has been created.

A global cache is created by the Write to Cache stage. Before you can use a global cache you must populate the cache with the data you want to look up. To do this, create a dataflow containing the **Write to Cache** stage.

Option Name	Description
Cache type	Select the Global cache option.
Cache name	Specifies the cache you want to query. To create a cache, use the Write to Cache stage.
Cache Fields	This column lists the fields in the cache. You cannot modify these field names.
Stage Fields	This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter a new name.
Type	This column lists the data type of each dataflow field.
Include	Check the box in this column to have the query return the value of the cache field. Clear the box if you do not want the query to return the cache field.

Option Name Description

Default Error Value

Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.
 You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data type Valid Default Error Value along with data type (in bracket)

Data type	Valid Default Error Value along with data type (in bracket)
Null	-1 (Integer) 1899-12-30 1899-12-30 12:00:00 (Time) 12:00:00 (Date) (Date Time)
False	False
Empty	Empty

Date	
Integer	✓
Long	✓
Float	✓
Big Decimal	✓
Double	✓
String	✓ ✓ ✓ ✓ ✓ ✓ ✓
Time	✓
Date Time	✓
Boolean	✓

Option Name	Description
Key Field	Specifies the field in the cache that will be used as a lookup key. If the value in the field in the Input Field column matches the value in the key field in the cache, then the query returns data from that record in the cache.
Input Field	Specifies the dataflow field, the value of which will be used as a key. If the value in this field matches the value in the key field in the cache, then the query returns data from that record in the cache.

Local Cache Options

A local cache is a temporary cache which is only used during the execution of Query Cache stage. The Query Cache builds the cache from the database table you choose. It then looks up data in the cache based on key fields and lookup conditions and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow.

A local cache is dynamic as it is created during a job execution of the Query Cache. Once Query Cache completes reading the data, the cache is automatically deleted from the memory. A local cache is recreated every time the Query Cache stage is executed. Choose a local cache if it is only going to be used in one dataflow or if the lookup table changes frequently.

Option name	Description						
Cache type	Specifies the Local cache option.						
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <table border="0"> <tr> <td>Connection name</td> <td>Enter a name for the connection. The name can be anything you choose.</td> </tr> <tr> <td>Database driver</td> <td>Select the appropriate database type.</td> </tr> <tr> <td>Connection options</td> <td>Specify the host, port, instance, user name, and password to use to connect to the database.</td> </tr> </table>	Connection name	Enter a name for the connection. The name can be anything you choose.	Database driver	Select the appropriate database type.	Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.
Connection name	Enter a name for the connection. The name can be anything you choose.						
Database driver	Select the appropriate database type.						
Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.						
Table/view	Specifies the table or view in the database that you want to query.						
Database Fields	This column lists the fields in the database. You cannot modify these field names.						

Option name	Description
Stage Fields	This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter the new name.
Type	This column lists the data type of each dataflow field.
Include	Check the box in this column to have the query return the value of the cache field. Clear the box if you do not want the query to return the cache field.

Option name Description

Default Error Value Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.

You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data type Valid Default Error Value along with data type (in bracket)

	Null	-1 (Integer)	1899-12-30 12:00:00 (Date)	1899-12-30 12:00:00 (Date Time)	12:00:00 (Time)	False	Empty
Date							
Integer		✓					
Long		✓					
Float		✓					
Big Decimal		✓					
Double		✓					
String	✓	✓	✓	✓	✓	✓	✓
Time					✓		
Date Time			✓				
Boolean						✓	

Option name	Description
Key Field	Specifies the field in the database that will be used as a look up key. If the value in the field in Input field column matches the value in the Key field in the database, then the query returns the data from that record in the database.
Input Field	Specifies the dataflow field whose value will be used as a key. If the value in this field matches the value in the Key field in the database, then the query returns data from that record in the database..

Advanced Cache Options

An advanced cache is a temporary cache similar to local cache. It is used during the execution of Query Cache stage. It builds the cache based on the SQL Query which reads the data from the tables mentioned in the query. It then looks up data in the cache based on the lookup keys mentioned in the where clause and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow..

An advanced cache is dynamic as it is created during a job execution of the Query Cache. Once Query Cache completes reading the data, the cache is automatically deleted from the memory. An advanced cache is recreated every time the Query Cache is executed. Choose an advanced cache option if it is going to read the data from multiple tables and there is some complex query needs to be executed for cache creation.

Option name	Description
Cache type	Specifies the Advanced cache option.
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection Name Enter a name for the connection. The name can be anything you choose.</p> <p>Database Driver Select the appropriate database type.</p> <p>Connection Options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Query	Provides SQL query to read data from the database. The query can read data from multiple tables.

Option name	Description
Where	This text is used as the where clause to lookup the cache created based on Query. User can specify input field in the Query using \$ operator as prefix. For example, <code>_id = \$_inputId</code> , Where <code>_inputId</code> is the input field and <code>_id</code> is the lookup column in the cache.
Get Fields	This populates the grid with the fields which are selected to be cached using SQL query.
Database Fields	This column lists the fields fetched in the database. You cannot modify these field names.
Stage Fields	This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter the new name.
Type	This column lists the data type of each dataflow field.

Option name Description

Default Error Value Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.

You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data type Valid Default Error Value along with data type (in bracket)

	Null	-1 (Integer)	1899-12-30 (Date)	1899-12-30 12:00:00 (Date Time)	12:00:00 (Time)	False	Empty
Date							
Integer		✓					
Long		✓					
Float		✓					
Big Decimal		✓					
Double		✓					
String	✓	✓	✓	✓	✓	✓	✓
Time					✓		
Date Time			✓				
Boolean						✓	

Runtime Tab

The options available in Runtime tab are common for global, local, and advanced caches.

Option Name	Description
Match options	<p>Specifies what to do if there is more than one record in the cache that matches the query. One of the following:</p> <p>Return all matches Return data from all records in the cache that have a matching value in the key field or fields.</p> <p>Return the first matching record Return data from only the first record in the cache that has a matching value in the key field or fields.</p> <p>Return the last matching record Return data from only the last record in the cache that has a matching value in the key field or fields.</p>
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Query DB

The Query DB stage allows you to use fields as parameters into a database query and return the results of the query as new fields in the dataflow.

Note: If you want to query a spatial database, use Query Spatial Data instead of Query DB.

General Tab

Option	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Specifies the table or view in the database that you want to query.
Where	<p>If you want to use a <code>WHERE</code> statement, enter it here. Note that you should not actually include the word <code>WHERE</code> in the statement. The purpose of a <code>WHERE</code> statement is to return only the data from records that match the condition you specify.</p> <p>To specify a value from a dataflow field, use this syntax:</p> <pre>\${<field name>}</pre> <p>Where <code><field name></code> is the name of a field in the dataflow.</p> <p>For example:</p> <pre>account_number=\${customer_key}</pre> <p>In this example, the query would return data from records where the value in the table column <code>account_number</code> matches the value in the dataflow field <code>customer_key</code>.</p> <p>Note: If you are querying a case-sensitive database, make sure you enter the field name in the same format as used in the database table. In other words, enclose the field name in quotes (") if the field names were quoted during table creation.</p> <p>Click Preview to see a preview of the data (first 50 records) based on the criteria you defined.</p> <p>Note: The preview feature in Query DB does not work if you use a dataflow field in the <code>WHERE</code> statement. Instead, you can preview the result using the dataflow inspection tool in Enterprise Designer.</p>

Option	Description
Return records with no results	Check this box if you want records whose queries return no results to still be returned by Query DB. If you clear this check box, the record will not be returned. We recommend that you leave this option checked.
Include	In the fields table, select the fields you want to include by clicking the Include box next to the field.

Sort Tab

If you want to sort records based on the value of a field, specify the fields you want to sort on.

Parameterizing Query DB at Runtime

You can configure the Query DB stage so that values in the `WHERE` clause are specified at runtime. This is useful in cases where you want to make the column name in the `WHERE` clause configurable using Dataflow Options.

1. Open the dataflow in Enterprise Designer.
2. Configure the **Connection** and **Table/View** name fields to point to the database you want to query.
3. In the **Where** field, enter a `WHERE` statement using the following format for values you want to parameterize: `${parameter}`.

For example:

```
${COL}=${EmployeeID}
```

Here `COL` represents a Dataflow option which will be populated with the column name for the table at runtime.

4. Close the Query DB options window.
5. Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** window appears.
6. Click **Add**. The **Define Dataflow Options** window appears.
7. Select the Query DB stage.
8. Specify **Option name** and **Option label**.

The value in the **Option name** field should be the same value as entered in the format `${parameter}` in the `WHERE` clause. In the **Option label** field, you can specify a different label or keep it the same as the **Option name**.

For example: `COL`

9. Specify the **Default value**. For example: "EmpID".
10. Click **OK**.

This procedure maps the actual database column name i.e. EmpID, to the runtime option name "COL". The database column name needs to be appropriately quoted with the specific quote identifier for the particular database.

Query NoSQL DB

The **Query NoSQL DB** stage allows lookups of required data from a NoSQL database. The stage supports MongoDB databases.

General Tab

Field Name	Description														
Connection	<p>Select the database connection you want to use.</p> <p>The available connections are the ones defined in Enterprise Designer, at Tools > NoSQL DB Connection Management.</p> <p>To create a new database connection, or modify or delete an existing database connection, click Manage. The NoSQL DB Connection Management window opens, the fields in which are:</p> <table border="0"> <tr> <td>Connection name</td> <td>Enter a name for the connection of your choice.</td> </tr> <tr> <td>NoSQL database</td> <td>Select the NoSQL database for which you are creating the connection.</td> </tr> <tr> <td>Username</td> <td>Enter the username to connect to the database.</td> </tr> <tr> <td>Password</td> <td>Enter the password to connect to the database.</td> </tr> <tr> <td>Hostname</td> <td>Specify the hostname on which the database runs.</td> </tr> <tr> <td>Port</td> <td>Specify the port to use to connect to the database.</td> </tr> <tr> <td>Database</td> <td>Specify the database from which the data is to be retrieved.</td> </tr> </table>	Connection name	Enter a name for the connection of your choice.	NoSQL database	Select the NoSQL database for which you are creating the connection.	Username	Enter the username to connect to the database.	Password	Enter the password to connect to the database.	Hostname	Specify the hostname on which the database runs.	Port	Specify the port to use to connect to the database.	Database	Specify the database from which the data is to be retrieved.
Connection name	Enter a name for the connection of your choice.														
NoSQL database	Select the NoSQL database for which you are creating the connection.														
Username	Enter the username to connect to the database.														
Password	Enter the password to connect to the database.														
Hostname	Specify the hostname on which the database runs.														
Port	Specify the port to use to connect to the database.														
Database	Specify the database from which the data is to be retrieved.														
Table/View	<p>Specifies the collection or view in the database that you want to query.</p> <p>Note: In a MongoDB database, a table/view is called a <i>collection</i>.</p>														

Field Name	Description
Schema File	<p>Click the Browse button (...) to select a JSON Schema file. This file is optional. Fields in the fields tab can be regenerated either using the schema file or database table/view.</p> <p>To clear the selected file path, click Clear.</p> <p>Note: Fields will always be generated using the schema file if one is selected.</p>
Where	<p>Enter the <code>where</code> clause to define the criteria for the lookup.</p> <p>For a list of the supported operators in the <code>WHERE</code> clause, see http://docs.mongodb.org/manual/reference/operator/query/.</p>
Preview	<p>Displays the records from the selected table or view.</p> <p>Note: Clicking Preview fetches the first 50 records from the selected database, without applying the filter criteria specified in the Where field.</p>
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to pass to the next stage. For more information, see [Defining Fields - Query NoSQL DB](#) on page 126.

Defining Fields - Query NoSQL DB

The **Fields** tab displays the field and its type as defined in the NoSQL DB/Schema file.

1. On the **Fields Tab**, click **Regenerate**.

This generates the aggregated data based on the first 50 records. The data is displayed in the following format: `Fieldname (datatype)`.

Note: If the schema file is browsed, the fields are generated using the schema file. The table/view is bypassed. To reset the schema file, click **Clear**.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.

- In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.

- You can also add additional fields that are not present in the table or schema file. Click **Add** to add a new field. To remove a field, click **Remove**.

Note: You can only add a new field under the List type.

- Click **OK**.

Configuring Dataflow Options - Query NoSQL DB

This procedure describes how to configure a dataflow to support runtime options for **Query NoSQL DB**.

- Open the dataflow in Enterprise Designer.
- If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
- Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
- Click **Add**. The **Define Dataflow Options** dialog box appears.
- Expand the **Query NoSQL DB** stage.
- The below dataflow options are exposed to query a Mongo DB database:
 - Connection
 - Table

The selected **Query NoSQL DB** option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at runtime in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Read From DB

The **Read From DB** stage reads data from a database table or view as input to a dataflow. The stage is available for jobs, services, and subflows but not for process flows.

General Tab

Field Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>

Field Name	Description
SQL	<p>Enter the SQL query to specify the records that need to be read from the data source on running the dataflow. You can manually type the SQL query in this field. Alternatively, use the Visual Query Builder to construct the query by clicking Build SQL....</p> <p>The SQL query can include variables instead of actual column names. Using variables allows you to customize the query at runtime. For more information, see Query Variables on page 133.</p>
Build SQL	<p>Create a complex query by selecting multiple columns, and creating joins and nested queries by clicking Build SQL. The Visual Query Builder opens. For more information, see Visual Query Builder on page 130.</p> <p>Note: A query created using the Visual Query Builder is displayed with fully qualified names of columns and tables in the SQL field.</p>
Regenerate Fields	<p>To see the schema of the data to be fetched by the query, click Regenerate Fields. If you edit an existing query, click Regenerate Fields to fetch the modified schema.</p> <p>Note: On clicking Regenerate Fields, the entity names in the SQL query are retained and not replaced with their fully qualified names.</p>
Preview	<p>To see a sample of the records fetched by the SQL query, click Preview.</p>

Note: The **Read From DB** stage allows you to modify the type of an input field.

Note: The **Read from DB** stage reads all values of the `date` datatype as `String` values. This is the behavior of the *jtDS driver*, which is the default driver used by Spectrum. To handle all `date` datatype values as is, use Microsoft's JDBC driver.

Runtime Tab

Field name	Description
Fetch size	<p>Select this option to specify the number of records to read from the database table at a time. For example, if the Fetch size value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.</p> <p>Setting an optimum fetch size can improve performance significantly.</p> <p>Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see Determining an Optimum Fetch Size on page 294.</p>
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Visual Query Builder

Visual Query Builder provides a visual interface for building complex SQL queries in the Read from DB stage. To work with Visual Query Builder, you need basic knowledge of SQL concepts.

To access Visual Query Builder, click the **Build SQL** button in Read from DB.

The query builder main window is divided into the following parts:

- The **Query Building Area** is the main area where the visual representation of query will be displayed. This area allows you to define source database objects, define links between them and configure properties of tables and links.
- The **Columns Pane** is located below the query building area. It is used to perform all necessary operations with query output columns and expressions. Here you can define field aliases, sorting and grouping, and define criteria.
- The page control above the query building area will allow you to switch between the main query and sub-queries.

Adding Objects to a Query

To add an object to a query, use the **Database Objects** tree. The objects within the tree are grouped by database, schema, and type. Drag the object you want to add to the query and drop it to the query building area. Alternatively, you can also double click the object to add it to the query.

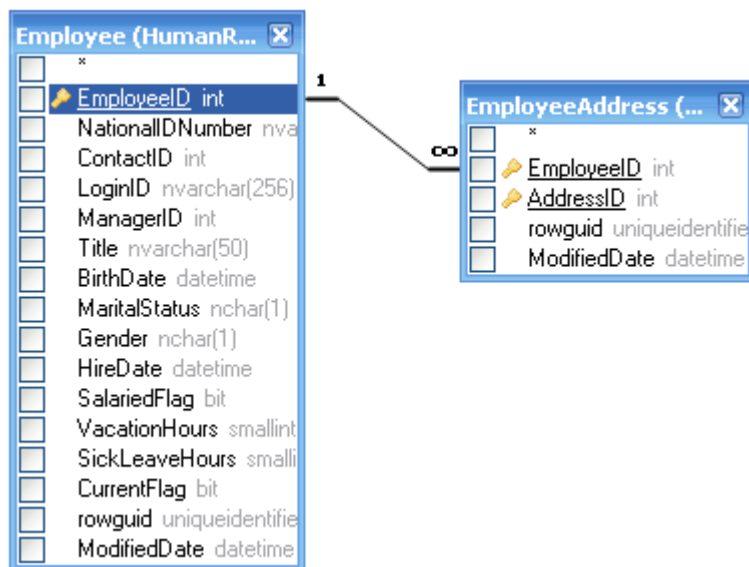
Setting Object Aliases

To set an alias for an object or derived table in the query, double-click the object and select **Properties**. The **Datasource Properties** dialog appears. It may contain other server-specific datasource options, but the Alias property is the same for all database servers.

Joining Tables

When two objects referenced with a foreign key relationship are added to the query, they become joined automatically with INNER JOIN. For those servers that do not support the JOIN clause, the query builder adds a condition to the WHERE part of the query.

To join two objects manually, select the field by which you want to link the object with another and drag it to the corresponding field of the other object. After you finish dragging, a line connecting the linked fields will appear. Key cardinality symbols are placed at the ends of link when a corresponding relationship exists in the database.



To remove a link between objects, double-click the link line and select **Remove**.

To change the join type, double click the link line.

Selecting Output Fields

To add a field to the list of query output fields, check the box at the left of the field name in the datasource field list in the **Query Building** area. To include all the fields of the object, check the

box at the left of the asterisk item in the datasource field list. You may also drag fields from the **Query Building** area to the **Columns** pane to get the same result.

If you do not select any fields from the query datasources, an asterisk item will be added to the select list of the resulting query ("Select * From ..."). This is because a SELECT query without any columns will produce an error for most database servers. The asterisk item is removed from the query if you select any field or add any output expression to the query.

Tip: Another way to add a field is to select a field name from the drop-down list of the Expression column in the **Columns** pane. You may also type any valid expression in the **Expression** column in the **Columns** pane. To insert an empty line to the **Columns** pane, press the Alt+Insert key.

To remove a field from the **Columns** pane, clear the check box at the left of the field name in the **Query Building** area or press the Alt+Delete key in the **Columns** pane.

To move a line up press the Alt+Up key. To move a line down press the Alt+Down key.

To remove an expression from the SELECT list of the query, clear the check box in the **Output** column.

To set an alias for an expression, enter the alias in the **Alias** column. Aliases become the headings of columns in the resulting dataset.

Sorting a Dataset

To sort the resulting dataset, use the Sort Type and Sort Order columns of the **Columns** pane. The Sort Type column allows you to sort in ascending or descending order. The Sort Order column allows you to set up the order in which fields will be sorted, if more than one field will be sorted.

To disable sorting by a field, clear the Sort Type column for the field.

Defining Criteria

To define criteria, use the **Criteria** column and the **Or** columns of the **Columns** pane. When writing conditions in these columns, omit the expression itself. For example, to get the following criteria in your query:

```
WHERE (Field1 >= 10) AND (Field1 <= 20)
```

Type the following in the Criteria cell of the Field1 expression:

```
>= 10 AND <= 20
```

Criteria placed in the **Or** columns will be grouped by columns using the AND operator and then concatenated in the WHERE (or HAVING) clause using the OR operator. For example, the criteria shown below will produce the SQL statement below. Please note that criteria for Field1 is placed both to the Criteria and Or columns.

Output	Expression	Aggregate	Alias	Sort Type	Sort Order	Grouping	Criteria	Or...	Or...
<input checked="" type="checkbox"/>	Field1					<input type="checkbox"/>	= 10	= 10	
<input checked="" type="checkbox"/>	Field2					<input type="checkbox"/>	< 0	> 10	
<input type="checkbox"/>						<input type="checkbox"/>			

```
WHERE (Field1= 10) AND ((Field2 < 0) OR (Field2 > 10))
```

Some expressions may be of Boolean type, for example the EXISTS clause. In this case you should type "= True" in the Criteria column of such expressions or "= False" if you want to place a NOT operator before the expression.

Grouping Output Fields

To build a query with grouping, mark expressions for grouping with the **Grouping** check box.

A query with grouping may have only grouping or aggregate expressions in the SELECT list. Thus the query builder allows you to set the Output check box for grouping and aggregate expressions. If you try to set this check box for a column without the grouping or aggregate function set, a Grouping check box will be set automatically to maintain the validity of resulting SQL query.

When the **Columns** pane contains columns marked with the **Grouping** check box, a new column called **Criteria for** appears in the grid. This column applies criteria to expression groups or to their values.

For example, you have a column "Quantity" with Aggregate function "Avg" in your query and you type > 10 in the Criteria column. Having the "for groups" value set in the Criteria for column, the resulting query will contain only groups with an average quantity greater than 10, and your query will have the "Avg(Quantity) > 10" condition in a HAVING clause. Having the "for values" value set in the Criteria for column, the result query will calculate the Average aggregate function only for records with Quantity value greater than 10, and your query will have the "Quantity > 10" condition in WHERE clause.

Defining SQL Query Properties

You can define options specific to your database server by using the context popup menu of the **Query Building** area.

Query Variables

In the **Read From DB** stage, while defining the query to be executed, you can include variables instead of actual column names. Using variables in the query allows you to customize the query conditions at runtime (using the **Dataflow Options**) or through the **Job Executor**.

However, you can also provide Stage Options value in the **Runtime** tab and view the schema and sample of records to be fetched by the query by using the **Regenerate Fields** and **Preview** buttons respectively.

A variable is defined using the format `#{variable}`, and inserted in either the `select` or `where` clause of an SQL query.

Note: You can edit a query generated using the **Visual Query Builder** to include variables. However, the edited query will not be readable by the Visual Query Builder anymore. The **Build SQL** button is disabled when you include a variable in a manually written or generated SQL query.

Inserting a Query Variable

1. Open the required job, which includes a **Read From DB** stage. Alternatively, add a **Read From DB** stage to the job.
2. Open the **Read From DB Options** of the **Read From DB** stage.
3. Create the SQL query in the **SQL** field, either manually or using the Visual Query Builder. For more information, see [Visual Query Builder](#) on page 130.
4. Add the desired conditions in the `where` clause of the query using variables with the syntax `#{variable}`.
For example, in a table `CUSTOMERS`, which has the column `AGE` with values such as 28, 32, 30, and so forth, and a column `SALARY` with values such as 1000, 1500, 2200, and so on, frame an SQL query as below:

```
select * from CUSTOMERS where #{condition1} > 28 and #{condition2} >
1200
```

Note: On inserting a variable in the `where` clause of the SQL query, the **Build SQL...** button is disabled.

5. To view the schema and the sample records to be fetched by the query, enter Stage Options value on the **Runtime** tab, and then click the **Regenerate Fields** and **Preview** buttons respectively.
6. Click **OK**.

The `where` clause of the SQL query can now be customized at runtime using the **Dataflow Options**, or while executing the job through the JobExecutor.

Note: A variable can be placed in the `select` clause of an SQL query as well. However, such a variable name should match the name of one of the columns of the table being queried.

Configuring a Query Variable as a Dataflow Option

1. Open the required job for which the query containing the variable(s) has been defined in a **Read From DB** stage.
2. Open **Edit > Dataflow Options...**
3. Click **Add**.
4. In the **Map dataflow options to stages** section, expand the **Read From DB** entry.

The variables defined in the SQL query in the **Read From DB** stage are listed along with the other attributes of the stage.

5. Select the variable you wish to customize using the corresponding checkbox.
6. Enter a relevant name for the variable in the **Option label** field.
7. In the **Default value** field, enter the column name which is to be used instead of the selected variable in the `where` clause of the SQL query. Alternatively, enter a constant value to be used instead of the variable in the `where` clause.

For example, for the below SQL query defined in the **Read From DB** stage:

```
select * from CUSTOMERS where #{condition1} > 28 and #{condition2} > 1200
```

you can select the column `AGE` of the table `CUSTOMERS` as the **Default value** for the variable `condition1`, and the column `SALARY` as the **Default value** for the variable `condition2`.

At runtime, the query is interpreted as:

```
select * from CUSTOMERS where AGE > 28 and SALARY > 1200
```

8. Repeat steps 5-7 for all the variables placed in the SQL query in the **Read From DB** stage.
9. Click **OK**.

On running the dataflow, the customized query is used to fetch the required data.

Configuring a Query Variable for Job Executor

Note: Ensure you have the Spectrum™ Job Executor downloaded to your server.

1. Create a text file that defines the default values of the variables included in the SQL query of the **Read From DB** stage of the job.

To assign a column name `AGE` as the default value to a variable `condition1`, create a text file, say `variables.txt`, and include the below line in the file:

```
condition1=AGE
```

To assign a constant value, say `20`, as the default value to a variable `condition1`, include the below line in the file:

```
condition1=20
```

2. While running the job using the Job Executor on the command prompt, use the argument `-o` followed by the path to the created text file.

For example, to run the job `ReadCustomerDataJob`, for which the default values of its variables have been defined in the text file `variables.txt`, run the below command on a command prompt:

```
java -jar jobexecutor.jar -h "localhost" -u "admin" -p "admin" -s
"8080" -j "ReadCustomerDataJob" -o "variables.txt"
```

On running the job using the Job Executor, the customized query is used to fetch the required data.

Note: For instructions and command-line syntax, see *Running a Job* in the Dataflow Designer Guide.

Read From File

The Read from File stage specifies an input file for a job or subflow. It is not available for services.

Note: If you want to use an XML file as input for your dataflow, use the Read from XML stage instead of Read from File. If you want to use a variable format file as input, use Read from Variable Format File.

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.

Field Name	Description
File name	<p>Specifies the path to the file. Click the ellipses button (...) to go to the file you want.</p> <p>You can read multiple files by using a wild card character to read data from multiple files in the directory. The wild card characters * and ? are supported. For example, you could specify *.CSV to read in all files with a .CSV extension located in the directory. In order to successfully read multiple files, each file must have the same layout (the same fields in the same positions). Any record that does not match the layout specified on the Fields tab will be treated as a malformed record.</p> <p>While reading a file from an HDFS file server, the compression formats supported are:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: The extension of the file indicates the compression format to be used to decompress the file.</p> <p>Attention: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Record type	<p>The format of the records in the file. Select one of:</p> <p>Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.</p> <p>Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.</p> <p>Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.</p>

Field Name	Description
Character encoding	The text file's encoding. Select one of these: <ul style="list-style-type: none"> UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. US-ASCII A character encoding based on the order of the English alphabet. UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first). UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first). ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1. ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3. ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5. CP850 An ASCII code page used to write Western European languages. CP500 An EBCDIC code page used to write Western European languages. Shift_JIS A character encoding for the Japanese language. MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions. CP1047 An EBCDIC code page with the full Latin-1 character set.

Field Name	Description						
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>						
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>						
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <table border="0"> <tr> <td>Unix (U+000A)</td> <td>A line feed character separates the records. This is the standard record separator for Unix systems.</td> </tr> <tr> <td>Macintosh (U+000D)</td> <td>A carriage return character separates the records. This is the standard record separator for Macintosh systems.</td> </tr> <tr> <td>Windows (U+000D U+000A)</td> <td>A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</td> </tr> </table> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>	Unix (U+000A)	A line feed character separates the records. This is the standard record separator for Unix systems.	Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.	Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.
Unix (U+000A)	A line feed character separates the records. This is the standard record separator for Unix systems.						
Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.						
Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.						

Field Name	Description
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum™ Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Record length	<p>For fixed width files, specifies the exact number of characters in each record.</p> <p>For line sequential files, specifies the length, in characters, of the longest record in the file.</p>
First row is header record	<p>Specifies whether the first record in a delimited file contains header information and not data.</p> <p>For example, this file snippet shows a header row in the first record.</p> <pre>"AddressLine1" "City" "StateProvince" "PostalCode" "7200 13TH ST" "MIAMI" "FL" "33144" "One Global View" "Troy" "NY" "12180"</pre>
Treat records with fewer fields than defined as malformed	<p>Delimited file records containing fewer fields than are defined on the Fields tab will be treated as malformed.</p>
Import	<p>Imports the file layout definition, encoding setting, and sort options from a settings file. The settings file is created by exporting settings from another Read from File or Write to File stage that used the same input file or a file that has the same layout as the file you are working with.</p>
Export	<p>Saves the file layout definition, encoding setting, and sort options to a settings file. You can then import these settings into other Read from File or Write to File stages that use the same input file or a file that has the same traits as the file you are working with now. You can also use the settings file with job executor to specify file settings at runtime.</p> <p>For information about the settings file, see The File Definition Settings File on page 147.</p>

Fields Tab

The Fields tab defines the names, positions, and, for fixed width and line sequential files, lengths of fields in the file. For more information, see these topics:

[Defining Fields In a Delimited Input File](#) on page 141

[Defining Fields In a Line Sequential or Fixed Width File](#) on page 144

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional. For more information, see [Sorting Input Records](#) on page 146.

Runtime Tab

Field Name	Description
File name	Displays the file name selected in the first tab.
Starting record	If you want to skip records at the beginning of the file when reading records into the dataflow, specify the first record you want to read. For example, if you want to skip the first 50 records, in a file, specify 51. The 51st record will be the first record read into the dataflow.
All records	Select this option if you want to read all records starting from the record specified in the Starting record field to the end of the file.
Max records	Select this option if you want to only read in a certain number of records starting from the record specified in the Starting record field. For example, if you want to read the first 100 records, select this option and enter 100.

Defining Fields In a Delimited Input File

The **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps on the **Fields** tab:

1. To define the fields already present in the input file, click **Regenerate**. Then, click **Detect Type**. This will automatically set the data type for each field based on the first 50 records in the file.
2. To add additional fields in the output, click **Add**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as `string` if you do not intend to perform any mathematical or date time operations with the data. However, if you intend to perform these

kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum™ Technology Platform supports these data types:

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum™ Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.

long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

5. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Management Console or Enterprise Designer will take effect.

- In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
- In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

6. In the **Position** field, enter the position of this field within the record.

For example, in this input file, AddressLine1 is in position 1, City is in position 2, StateProvince is in position 3, and PostalCode is in position 4.

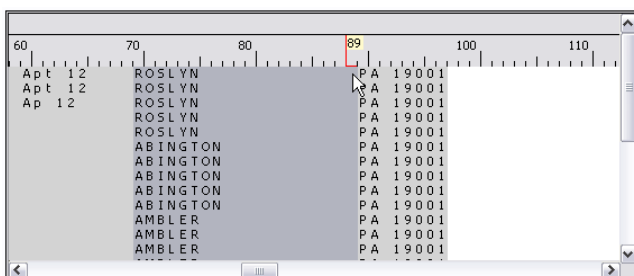
```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|"12180"
```

- If you want to have any excess space characters removed from the beginning and end of a field's value string, select the **Trim** check box.

Defining Fields In a Line Sequential or Fixed Width File

In the Read from File stage, the **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

- On the **Fields** tab, under **Preview**, click at the beginning of a field and drag to the left so that the desired field is highlighted, as shown here:



- In the **Name** field, enter the field you want to add.
- In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical or date/time operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum™ Technology Platform supports the following data types:

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre data-bbox="428 779 1424 934"> <Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names> </pre> <p>It is important to note that the Spectrum™ Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.</p>
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

- To apply the `packed decimal` storage format to the field, check the **Packed Decimal** checkbox. The `packed decimal` type uses 4 bits as compared to the `integer` type which uses 8 bits.

Note: This storage format is available only on selecting the datatypes `double`, `integer` and `long` while reading line sequential and fixed width files.

- If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Management Console or Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
- b) In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

6. The **Start Position** and **Length** fields are automatically filled in based on the selection you made in the file preview.
7. If you want to have any excess space characters removed from the beginning and end of a field's character string, select the **Trim Spaces** check box.
8. Click **OK**.

Sorting Input Records

In the Read from File stage, the **Sort Fields** tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional.

1. On the **Sort Fields** tab, click **Add**.
2. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields defined in this input file.
3. In the **Order** column, select Ascending or Descending.
4. Repeat until you have added all the input fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.
5. Default sort performance options for your system are set in Management Console. If you want to override your system's default sort performance options, click **Advanced**. The **Advanced Options** dialog box contains the following sort performance options:

In memory record limit Specifies the maximum number of data rows a sorter will hold in memory before it starts paging to disk. By default, a sort of 10,000 records or less will be done in memory and a sort of more than 10,000 records will be performed as a disk sort. The maximum limit is 100,000 records. Typically an in-memory sort is much faster than a disk sort, so this value should be set high enough so that most of the sorts will be in-memory sorts and only large sets will be written to disk.

Note: Be careful in environments where there are jobs running concurrently because increasing the **In memory record limit** setting increases the likelihood of running out of memory.

Maximum number of temporary files Specifies the maximum number of temporary files that may be used by a sort process. Using a larger number of temporary files can result in better performance. However, the optimal number is highly dependent on the configuration of the server running Spectrum™ Technology Platform. You should experiment with different settings, observing the effect on performance of using more or fewer temporary files. To calculate the approximate number of temporary files that may be needed, use this equation:

$$\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFiles$$

Note that the maximum number of temporary files cannot be more than 1,000.

Enable compression Specifies that temporary files are compressed when they are written to disk.

Note: The optimal sort performance settings depends on your server's hardware configuration. Nevertheless, the following equation generally produces good sort performance:

$$\frac{(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2)}{TotalNumberOfRecords} \geq$$

The File Definition Settings File

A file definition settings file contains the file layout, encoding, and sort options that have been exported from a Read from File or Write to File stage. The file definitions settings file can be imported into Read from File or Write to File to quickly set the stage's options instead of manually specifying the options.

The easiest way to create a file definition settings file is to use specify the file settings using Read from File or Write to File, then click the **Export** button to generate the file definitions settings file.

However, for your information the schema of the file definition settings file is shown below. Each element in the XML file has a type, and if that type is anything other than string or integer, the acceptable values are shown. These values correspond directly to options in the stage's dialog box. For example, the FileTypeEnum element corresponds to the Record Type field on the File Properties tab, and the following three values are shown in the schema: linesequential, fixedwidth, and delimited.

Note: If you enter "custom" for the LineSeparator, FieldSeparator or TextQualifier fields, a corresponding custom element must also be included (for example, "CustomLineSeparator", "CustomFieldSeparator", or "CustomTextQualifier") with a hexadecimal number representing the character, or sequence of characters, to use.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FileSchema" nillable="true" type="FileSchema"/>
  <xs:complexType name="FileSchema">
    <xs:sequence>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="linesequential"
        name="Type"
        type="FileTypeEnum"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="UTF-8" name="Encoding" type="xs:string"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="RecordLength"
        type="xs:int"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="default"
        name="LineSeparator"
        type="LineSeparatorEnum"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="CustomLineSeparator"
        type="xs:string"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="comma"
        name="FieldSeparator"
```

```

        type="FieldSeparatorEnum"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="CustomFieldSeparator"
  type="xs:string"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="none"
  name="TextQualifier"
  type="TextQualifierEnum"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="CustomTextQualifier"
  type="xs:string"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="false"
  name="HasHeader"
  type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="true"
  name="EnforceColumnCount"
  type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="Fields"
  type="ArrayOfFieldSchema"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="FileTypeEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="linesequential"/>
    <xs:enumeration value="fixedwidth"/>
    <xs:enumeration value="delimited"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="LineSeparatorEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="default"/>
    <xs:enumeration value="windows"/>
    <xs:enumeration value="unix"/>
    <xs:enumeration value="mac"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FieldSeparatorEnum">

```

```

<xs:restriction base="xs:string">
  <xs:enumeration value="comma"/>
  <xs:enumeration value="tab"/>
  <xs:enumeration value="space"/>
  <xs:enumeration value="semicolon"/>
  <xs:enumeration value="period"/>
  <xs:enumeration value="pipe"/>
  <xs:enumeration value="custom"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="TextQualifierEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="single"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ArrayOfFieldSchema">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="unbounded"
      name="Field"
      nillable="true"
      type="FieldSchema"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="FieldSchema">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Name"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      default="string"
      name="Type"
      type="xs:string"/>
    <xs:element
      minOccurs="1"
      maxOccurs="1"
      name="Position"
      type="xs:int"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Length"
      type="xs:int"/>
    <xs:element
      minOccurs="0"

```

```

        maxOccurs="1"
        default="false"
        name="Trim"
        type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="Locale"
  type="Locale"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="Pattern"
  type="xs:string"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="none"
  name="Order"
  type="SortOrderEnum"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Locale">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Country"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Language"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Variant"
      type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="SortOrderEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="ascending"/>
    <xs:enumeration value="descending"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Configuring Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for Read from File stage.

1. Open the dataflow in Enterprise Designer.
2. If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
3. Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
4. Click **Add**. The **Define Dataflow Options** dialog box appears.
5. Expand the **Read from File** stage.

The Dataflow options exposed are:

1. Character Encoding
 2. Field Separator
 3. Text Qualifier
 4. Record Length
 5. First Row is Header Record
 6. Starting Record
 7. Max Records
6. The selected Read from File option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.
 7. Enter a description of the option in the **Description** field.
 8. In the **Target** field, select the option **Selected stage(s)**.
 9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
 10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Dataflow Options Rules

1. *Character Encoding*: All encoding types that are valid for the underlying JVM are accepted. This option cannot be empty.
2. *Field Separator*: Any single character delimiter is accepted. Currently, HEX values and spaces are not supported.
3. *Text Qualifier*: Any single character qualifier is accepted. HEX values are not supported.
4. *Record Length*: Only integers accepted. Cannot be blank or non numeric.
5. *Starting Record*: Only integers accepted. Cannot be non numeric.
6. *Max records*: Only integers accepted. Cannot be non numeric.
7. *First Row is Header*: Only boolean values of `true` and `false` accepted. Cannot be blank.

Read from Hadoop Sequence File

The Read from Hadoop Sequence File stage reads data from a sequence file as input to a dataflow. A sequence file is a flat file consisting of binary key/value pairs. For more information, go to wiki.apache.org/hadoop/SequenceFile.

Note: The Read from Hadoop Sequence File stage only supports delimited, uncompressed sequence files located on Hadoop Distributed File System (HDFS).

File Properties Tab

Fields	Description
Server	Indicates the file you select in the File name field is located on the Hadoop system. You need to create a connection to the Hadoop file server in the Management Console before using it in the stage. If you select a file on the Hadoop system, the server name will be the name you specify in the Management Console while creating a file server.
File name	Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.

Fields	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier:</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>

Fields Tab

The Fields tab defines the names, positions, and types of fields in the file. For more information, see [Defining Fields In an Input Sequence File](#) on page 155.

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional. For more information, see [Sorting Input Records](#) on page 156.

Filter Tab

The Field tab defines fields by which to filter the input records before they are sent into the dataflow. For more information, see [Filtering Input Records](#) on page 156.

Defining Fields In an Input Sequence File

In the Read from Hadoop Sequence File stage, the **Fields** tab defines the names, positions, and types of fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps on the **Fields** tab:

1. To define the fields already present in the input file, click **Regenerate**. Then, click **Detect Type**. This will automatically set the data type for each field based on the first 50 records in the file.
2. To add additional fields in the output, click **Add**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).

string A sequence of characters.

5. In the **Position** field, enter the position of this field within the record.

For example, in this input file, AddressLine1 is in position 1, City is in position 2, StateProvince is in position 3, and PostalCode is in position 4.

```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|"12180"
```

6. If you want to have any excess space characters removed from the beginning and end of a field's value string, select the **Trim** check box.

Sorting Input Records

In the Read from Hadoop Sequence File stage, the **Sort Fields** tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional.

1. In Read from Hadoop Sequence File, click the **Sort Fields** tab.
2. On the **Sort Fields** tab, click **Add**.
3. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields defined in this input file.
4. In the **Order** column, select Ascending or Descending.
5. Repeat until you have added all the input fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.

Filtering Input Records

In the Read from Hadoop Sequence File stage, the **Filter** tab defines fields by which to filter the input records before they are sent into the dataflow. Filtering is optional.

1. In Read from Hadoop Sequence File, click the **Filter** tab.
2. In the **Combine expression method** field, choose **All** if you want all the expressions to evaluate to true in order for the record to be routed to this port; select **Any** if you want records to be routed to this port if one or more of the expressions is true.
3. Click **Add**, specify the field to test, the operator, and a value. The operators are listed in the following table.

Operator	Description
Is Equal	Checks if the value in the field matches the value specified.
Is Not Equal	Checks if the value in the field does not match the value specified.
Is Greater Than	Checks if the field has a numeric value that is greater than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Greater Than Or Equal To	Checks if the field has a numeric value that is greater than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.

Operator	Description
Is Less Than	Checks if the field has a numeric value that is less than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Less Than Or Equal To	Checks if the field has a numeric value that is less than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Null	Checks if the field is a null value.
Is Not Null	Checks if the field is not a null value.

4. Select the **Trim** option as desired. This option, first trims all the white spaces that may be present before and after the value of the field, before filtering the data in the field.
5. Repeat until you have added all the input fields you wish to use for filtering.

Read From Hive File

The **Read from Hive File** stage reads data from the selected file, which can be in any of the following formats:

- ORC
- RC
- Parquet
- Avro

File Properties tab

Fields	Description
Server	Indicates the file you select in the File name field is located on the Hadoop system. You need to create a connection to the Hadoop file server in the Management Console before using it in the stage. If you select a file on the Hadoop system, the server name will be the name you specify in the Management Console while creating a file server.

Fields	Description
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.</p> <p>Note: The schema of an input file is imported as soon as you browse to the correct location and select the file. This imported schema cannot be edited. You may, however, rename the columns of the schema as required.</p> <p>The first 50 records of the file are fetched in the Preview grid on selecting the file.</p>
File type	<p>Select the type of the file being read:</p> <ul style="list-style-type: none"> • ORC • RC • Parquet • Avro

Note: In case of RC files, to generate the Preview, define the schema in the **Fields** tab and then click **Preview** in the **File Properties** tab.

Fields tab

The **Fields** tab defines the names, datatypes, positions of the fields as present in the input file, as well as the user-given names for the fields. For more information, see [Defining Fields for Reading from Hive File](#) on page 158.

Defining Fields for Reading from Hive File

In the **Fields** tab of the **Read from Hive File** stage, the schema names, datatypes, positions, and the given names of the fields in the file are listed.

1. Click **Regenerate**.

In case of ORC, Avro, and Parquet files, this generates the schema based on the metadata of the existing file. In case of RC files, any fields added before clicking **Preview** are cleared.

The grid displays the columns **Name**, **Type**, **Stage Field**, and **Include**.

The **Name** column displays the field name, as derived from the header record of the file.

The **Type** column lists the datatypes of each respective field of the file.

The stage supports the following data types:

boolean	A logical type with two values: true and false.
date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

Note: The `datetime` datatype in Spectrum maps to the `timestamp` datatype of Hive files.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The `bigdecimal` data type supports more precise calculations than the `double` data type.

Note: For RC, Avro, and Parquet Hive files, fields of the `decimal` datatype in the input file are converted to `bigdecimal` datatype.

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).

Note: The `long` datatype in Spectrum maps to the `bigint` datatype of Hive files.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

string A sequence of characters.

Note: In case of RC files, `smallint` and complex datatypes are not supported.

The **Position** column displays the starting position of the respective field within a record.

- In the **Stage Field** column, edit the existing field name to the desired name for each field. By default, this column displays the field names read from the file.
- In the **Include** column, select the checkboxes against the fields you wish to include in the output of the stage. By default, all the fields are selected in this column.
- For RC files, you can add and remove fields, and modify the sequence of the selected columns in the output using the below buttons:

Option Name	Description
Add	Adds a field to the output.
Modify	Modifies the selected field's name and datatype.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the position of the selected field in the output.

Note: This feature is only available for RC files.

5. Click **OK**.

Read from HL7 File

The **Read from HL7 File** stage reads Health Level Seven (HL7) data from a text file as input to a dataflow. HL7 is a messaging standard used in the healthcare industry to exchange data between systems. For more information about HL7, go to www.hl7.org.

HL7 Message Format

Data in an HL7 message is organized hierarchically as follows:

- message
 - segment
 - field
 - component
 - Subcomponent

Each line of an HL7 message is a segment. A *segment* is a logical grouping of fields. The first three characters in a segment identify the segment type. In the above message, there are five segments

MSH (message header), PID (patient identification), two NK1 (next of kin) segments, and IN1 (insurance).

Each segment consists of fields. A *field* contains information related to the purpose of the segment, such as the name of the insurance company in the IN1 (insurance) segment. Fields are typically (but not always) delimited by a | character.

Fields can be divided into *components*. Components are typically indicated by a ^ character. In the above example, the PID (patient identification) segment contains a patient name field containing LEVERKUHN^ADRIAN^C which has three parts, last name (LEVERKUHN), first name (ADRIAN), and middle initial (C). Components can be divided into *subcomponents*. Subcomponents are typically indicated by a & character.

The following is an example of an HL7 message:

```
MSH|^~\&||.||||199908180016||ADT^A04|ADT.1.1698593|P|2.7
PID|1||000395122||LEVERKUHN^ADRIAN^C||19880517180606|M|||6 66TH AVE
NE^^WEIMAR^DL^98052|| (157) 983-3296|||S||12354768|87654321
NK1|1|TALLIS^THOMAS^C|GRANDFATHER|12914 SPEM
ST^^ALIUM^IN^98052| (157) 883-6176
NK1|2|WEBERN^ANTON|SON|12 STRASSE MUSIK^^VIENNA^AUS^11212| (123) 456-7890
IN1|1|PRE2||LIFE PRUDENT BUYER|PO BOX
23523^WELLINGTON^ON^98111|||19601|||THOMAS^JAMES^M|F|||||||||||||ZKA535529776
```

Note: To create an HL7 file using the given sample text:

1. Copy and paste the sample text into a new document using any text editing software, say Notepad++.
2. Make the required content changes.
3. Configure the settings to view EOL (End of Line) on the text. In Notepad++, go to **View > Show Symbol > Show End of Line**.
4. Change the EOL (End of Line) Conversion format to CR (Carriage Return). In Notepad++, go to **Edit > EOL Conversion > Old Mac Format**.
5. Save the HL7 file after making this format change.

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.

Field Name	Description
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.</p> <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
HL7 Version	<p>The version of the HL7 standard used in the file you specified. For example, "2.7" means that the file uses HL7 version 2.7. The HL7 version is indicated by the 12th field in the MSH segment.</p>

Field Name	Description
Character encoding	The text file's encoding. Select one of these:
UTF-8	Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
UTF-16	Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
US-ASCII	A character encoding based on the order of the English alphabet.
UTF-16BE	UTF-16 encoding with big endian byte serialization (most significant byte first).
UTF-16LE	UTF-16 encoding with little endian byte serialization (least significant byte first).
ISO-8859-1	An ASCII character encoding typically used for Western European languages. Also known as Latin-1.
ISO-8859-3	An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.
ISO-8859-9	An ASCII character encoding typically used for Turkish language. Also known as Latin-5.
CP850	An ASCII code page used to write Western European languages.
CP500	An EBCDIC code page used to write Western European languages.
Shift_JIS	A character encoding for the Japanese language.
MS932	A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.
CP1047	An EBCDIC code page with the full Latin-1 character set.

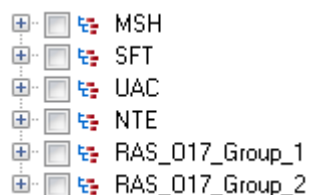
Field Name	Description
Validate	<p data-bbox="553 373 1430 520">These options specify whether to check the file to ensure that it conforms to the HL7 2.7 standard. If any message in the file fails validation, it is treated as a malformed record and the malformed record options specified for the job (in Enterprise Designer under Edit > Job Options) or for the system (in Management Console) will take effect.</p> <p data-bbox="553 541 1430 632">Required fields Check this box if you want to make sure that each segment, field, component, and subcomponent contains the elements that are required by the HL7 2.7 standard.</p> <p data-bbox="553 653 1430 743">Length Check this box if you want to make sure that each element meets the minimum and maximum length requirements for the element as defined in the HL7 2.7 standard.</p>

Field Name	Description
Ignore unexpected	<p>Select these options if you want to allow messages to contain segments, fields, components, and subcomponents that are not in the expected position. The expected positions are defined in the HL7 standard or, in the case of custom message types, in the HL7 Schema Management tool in Enterprise Designer.</p> <p>For example, consider the following custom message schema:</p> <pre data-bbox="552 556 649 756">MSH [PID] {ZSS} PV1 NK1 {[DG1]}</pre> <p>And this data:</p> <pre data-bbox="552 829 1421 1029">MSH ^~\& Pharm GenHosp CIS GenHosp 198807050000 RAS^O17 RAS1234 P 2.7 ZSS 100 abc PID 1234 PATID1234^5^M11^ADT1^MR^GOOD HEALTH HOSPITAL~123456789^^^USSSA^SS PV1 O O/R 0148^ADDISON, JAMES 0148^ADDISON, JAMES NK1 Son John</pre> <p>In this case the segment <code>PID</code> is unexpected because it is before the <code>ZSS</code> segment. Messages that contain elements in unexpected positions are treated as malformed records and the malformed record options specified for the job (in Enterprise Designer under Edit > Job Options) or for the system (in Management Console) take effect. By default all the Ignore unexpected options are enabled to allow as many records as possible to be processed successfully.</p> <p>Segments Check this box to allow messages to contain segments that are not defined in the HL7 2.7 standard. Unexpected segments are ignored and other segments in the message are processed.</p> <p>Fields Check this box to allow segments to contain fields that are not defined in the HL7 2.7 standard. Unexpected fields are ignored and other fields in the segment are processed.</p> <p>Components Check this box to allow fields to contain components that are not defined in the HL7 2.7 standard. Unexpected components are ignored and other components in the field are processed.</p> <p>Subcomponents Check this box to allow components to contain subcomponents that are not defined in the HL7 2.7 standard. Unexpected subcomponents in the component are ignored and other subcomponents are processed.</p>

Fields Tab

The **Fields** tab displays the segments, fields, components, and subcomponents. Use the **Fields** tab to select the data you want to read into the dataflow.

Segment groups, which are collections of segments that are used together to contain a category of data, are displayed using a numbering system that shows where in the message schema the group appears. Each segment group is given a label "Group_n" where "n" is a number that corresponds to the group's position in the message schema. To illustrate how the number system works, consider this example:

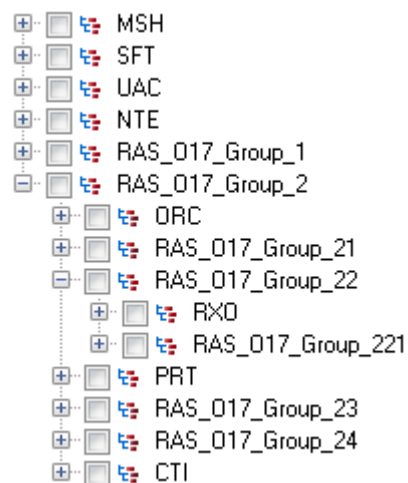


This example shows the field list for the message RAS^017. This message has two segment groups: RAS_017_Group_1 and RAS_017_Group_2. The "Group_1" segment group refers to the first segment group in the RAS^017 schema and the second group, "Group_2", refers to the second group listed in the RAS^017 schema.

To determine which segment group is represented by "Group_1" and "Group_2", find the description of the message RAS^017 in the document *HL7 Version 2.7 Standard*. You can download a copy of this document from www.hl7.org.

In the description of the message, find the first group, which in the case of RAS^017 is the PATIENT group. The second group in the schema is the ORDER group.

Segment groups that are nested under a segment group have an additional number appended to their group number. For example, Group_21 represents the first group nested under the second group. Further subgroups have additional numbers appended to them, such as Group_221, which for message RAS^017 represents the segment group ORDER_DETAIL_SUPPLEMENT. An example of nested groups is shown here:



The controls on the **Fields** tab are described in the following table.

Table 3: Fields Tab

Option Name	Description
Regenerate	<p>Click this button to populate the Fields tab with a list of all segments, fields, components, and subcomponents for the message type contained in the input file. All elements for the message type are displayed based on the HL7 schema regardless of whether the input file contains all the elements. For example, if the file contains an RAS message, the schema for the entire RAS message type will be displayed regardless of whether the input file actually contains data for all the segments, fields, components, and subcomponents.</p> <p>If you have defined any custom elements using the HL7 Schema Management tool in Enterprise Designer, those elements are also listed.</p>
Expand All	Expands all elements in the fields tab so you can view all segments, fields, components, and subcomponents of the message types contained in the file.
Collapse All	Closes all nodes in the view so that only the segments are displayed. Use this to easily see the segments for the message types in the file. You can then expand individual segments in order to view the fields, components, and subcomponents in a segment.
Select all	Check this box to create dataflow fields for all segments, fields, components, and subcomponents for all message types included in the file.

Flattening HL7 Data

HL7 data is organized hierarchically. Since many stages require data to be in a flat format, you may have to flatten the data in order to make the data usable by downstream stages for things like address validation or geocoding.

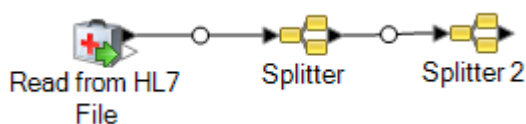
The following procedure describes how to use a Splitter stage to flatten HL7 data.

1. Add a Read from HL7 File stage to your data flow and configure the stage.
2. Add a Splitter stage and connect it to Read from HL7 File.
3. Add additional Splitter stages as needed so that you have one splitter stage for each segment, field, or component that you want to flatten.

Note: You only need to flatten the data that you want to process in a downstream stage. Other data can remain in hierarchical format. For example, if you only want to process address data, you only need to flatten the address data.

4. Connect all the Splitter stages.

You should now have a data flow that looks like this:



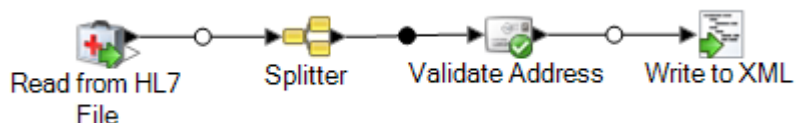
5. Double-click the first Splitter stage to open the stage options.
6. In the **Split at** field, select the segment, field, or component that you want to flatten.
7. Click **OK**.
8. Configure each additional Splitter stage, selecting a different segment, field, or component in each Splitter's **Split at** field.
9. Add additional stages as needed after the last Splitter to complete your dataflow.

Example

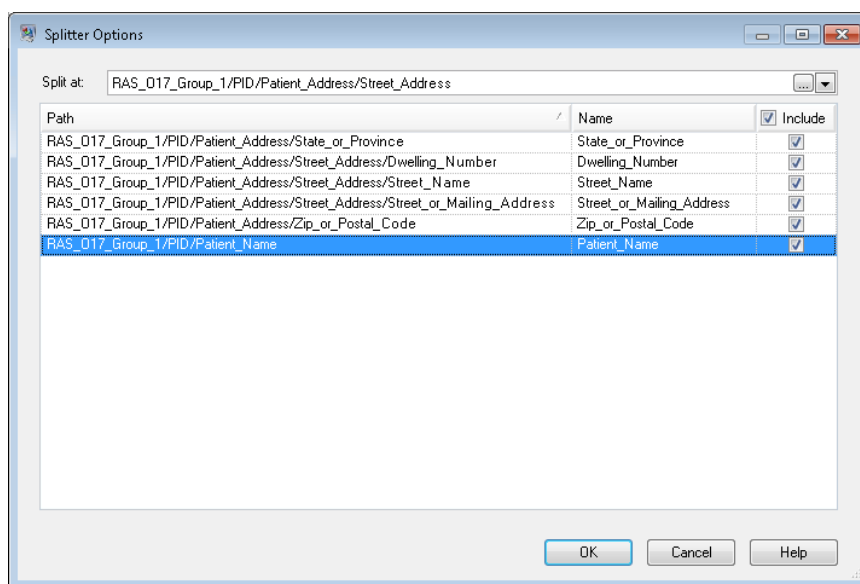
You have the following HL7 data and you want to validate the address in the PID segment.

```
MSH|^~\&|||.|||199908180016||RAS^O17|ADT.1.1698594|P|2.7
PID|1||000395122||SMITH^JOHN^D||19880517180606|M|||One Global
View^^Troy^NY^12180||(630)123-4567|||S||12354768|87654321
```

In order to do this you need to convert that address data to flat data so that it can be processed by the Validate Address stage. So, you create a dataflow containing a Splitter followed by a Validate Address stage, like this:



The Splitter stage is configured to split at the PID/Patient_Address/Street_Address component, which converts this data to flat data.



The channel that connects the Splitter stage to the Validate Address stage renames the fields to use the field names required by Validate Address:

Street_or_Mailing_Addres is renamed to AddressLine1, State_or_Province is renamed to StateProvince, and Zip_or_Postal_Code is renamed to PostalCode.

In this example, the output is written to an XML file containing this data.

```
<?xml version='1.0' encoding='UTF-8'?>
<XmlRoot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <PatientInformation>
    <Confidence>95</Confidence>
    <RecordType>Normal</RecordType>
    <CountryLevel>A</CountryLevel>
    <ProcessedBy>USA</ProcessedBy>
    <MatchScore>0</MatchScore>
    <AddressLine1>1 Global Vw</AddressLine1>
    <City>Troy</City>
    <StateProvince>NY</StateProvince>
    <PostalCode>12180-8371</PostalCode>
    <PostalCode.Base>12180</PostalCode.Base>
    <PostalCode.AddOn>8371</PostalCode.AddOn>
    <Country>United States Of America</Country>
    <Patient_Name>
      <Family_Name>
        <Surname>SMITH</Surname>
      </Family_Name>
      <Given_Name>JOHN</Given_Name>
    <Second_and_Further_Given_Names_or_Initials_Thereof>
      D
    </Second_and_Further_Given_Names_or_Initials_Thereof>
  </PatientInformation>
</XmlRoot>
```

```

</Second_and_Further_Given_Names_or_Initials_Thereof>
  </Patient_Name>
</PatientInformation>
</XmlRoot>

```

Adding a Custom HL7 Message

The Read from HL7 File stage validates messages using the HL7 2.7 schema. However, your HL7 data may contain messages that are not part of the HL7 standard. If you want the Read from HL7 File stage to validate your customized HL7 data, you need to create a custom HL7 schema. This topic describes how to create a custom HL7 schema using the HL7 Schema Management tool. For more information about HL7, go to www.hl7.org.

1. In Enterprise Designer, go to **Tools > HL7 Schema Management**.

This will open the HL7 Schema Management window which contains a list of supported messages. These messages are predefined by HL7.

2. In the **HL7 Schema Management** window, click **Add**.
3. In the **Message type** field, specify a type for a custom HL7 message.

The message type indicates what health-related information is being provided in the message. For example, an ADT (Admin Discharge Transfer) message type is used to exchange the patient state within a healthcare facility and an ORU (Observation Result) message type is used to transmit observations and results from the LIS (Lab Information System) to the HIS (Hospital Information System).

4. In the **Trigger event** field, specify an event code.

The trigger event is a real-world event that initiates communication and the sending of a message. Both the message type and trigger event are found in the MSH-9 field of the message. For example, MSH-9 field might contain the value ADT^A01. This means that ADT is the HL7 message type, and A01 is the trigger event.

5. In the **Description** field, type a description for a custom HL7 message.

This field helps you to understand more about a message type. For example, if you are adding a XYZ message type, you can provide a description that it is used to exchange the patient state within a healthcare facility.

You will now see a newly created message under the **Definition**. Click on the plus sign to expand the message. You can see the MSH segment is added automatically.

6. To add an existing segment to a message
 - a) Click **Select Segment**.

- b) Select the segments you want to add to the message and click **OK**.

A schema of a selected segment is displayed in the **Segment Schema** grid and the checked messages are added in the message schema.

- 7. To add a custom segment to a message

- a) Click **Select Segment**.
- b) Click **Add Segment**.
- c) In the **Name** field, specify a name for the segment and click **OK**.

The newly added segment is displayed at the bottom of the **Segments Supported** list.

- d) Select the added custom segment and then click the **Add field** button.
- e) In the **Name** field, specify a field name of the selected segment.

For example, a PID (Patient information) segment contains field names as Patient ID, Patient Name, Patient Address, County Code, and so on.

- f) In the **Type** field, select an appropriate data type.

HL7 data types define the kind of data that can be included in a field, and are used throughout the HL7 message structure. For example, ST for string, TX for text data and FT for formatted data.

- g) In the **Normative length** field, specify the minimum and maximum length of the field using the following format: m..n. You can also specify a list of possible values for a length of the field using the following format: x,y,z.

For example, length of 1..3 means the length of the item may be either 1, 2 or 3 and the length of 1, 3, 4 means the length of the item may be either 1, 3 or 4 but not 2. A value other than 1, 3, and 4 would be treated as invalid.

- h) In the **Optionality** field, specify whether a field is optional or required.

O

The field is optional.

R

The field is required.

- i) In the **Repetition** field, if you want to allow the field to appear more than once in the segment, check the **Repetition** box and specify the number of times the field can be used.

For example, a value of 3 would mean that the field can have three occurrences. If it is unspecified, there will only be one occurrence, which means this field will not be repeated.

- 8. Click **OK**.

You can also choose the options **Optional** and **Repeating** from the Segment properties.

- 9. Choose **Optional** to make the selected segment as optional and choose **Repeating** to allow a repetition of a selected segment in a message.

10. Click **OK**.

The newly added message is displayed at the bottom of the list.

Read from NoSQL DB

The Read from NoSQL DB stage reads data from a database table as input to a dataflow. The stage supports the MongoDB and Couchbase database types.

General Tab

Field Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the NoSQL Connection Management in the Tools menu of Enterprise Designer. If you want to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>NoSQL database Select the appropriate database type.</p> <p>Username Enter the username to connect to the database.</p> <p>Note: For Couchbase, username is not mandatory. You can enter any username. The connection will be successful as long as you use the correct password that you supplied when creating the bucket.</p> <p>Password Enter the password to connect to the database.</p> <p>Hostname Specify the hostname on which the database runs.</p> <p>Port Specify the port to use to connect to the database.</p> <p>Database Specify the database from which the data is to be retrieved.</p> <p>Note: While the term Database is used in the user interface, Couchbase calls it a bucket.</p>
Table/View	<p>Specifies the collection or view in the database that you want to query.</p> <p>Note: While the term Table/View is used in the user interface, MongoDB calls it a <i>collection</i>, and Couchbase calls it a <i>view</i>.</p>

Field Name	Description
Schema File	<p>Click the Browse button (...) to select a JSON Schema file. This file is optional. Fields in the fields tab can be regenerated either using the schema file or database table/view.</p> <p>To clear the selected file path, click Clear.</p> <p>Note: Fields will always be generated using the schema file if one is selected.</p>
Where	<p>Enter the required filter criteria, if any, using MongoDB syntax, to fetch specific records. Leave the field blank if no filter criteria is required.</p> <p>The following syntax is for a clause with an <i>equal to</i> operator:</p> <pre>{ "<column name>" : "<filter value>" }</pre> <p>You can join multiple clauses using the required operators. For a list of the supported operators in the <i>where</i> clause, see http://docs.mongodb.org/manual/reference/operator/query/.</p> <p>For example, to fetch records where the value of the column <code>customer_name</code> matches the value <code>John</code>, and the value of the column <code>customer_age</code> is greater than or equal to 45, enter the below:</p> <pre>{ \$and: [{ "customer_name": "John" }, { \$gte: ["customer_age", "45"] }] }</pre> <p>Attention: Ensure you do not include the keyword <code>where</code> in this field.</p> <p>Note: Currently, this field is visible only on selecting a MongoDB connection.</p>
Ignore Absent Fields	<p>Fields defined in the schema, if not present in the actual record will not flow to the next stage if this option is selected.</p> <p>Note: If you do not enable this option, fields that are not present in the database table or view, are added and processed with the value as NULL.</p>
Preview	<p>Displays the records from the selected table.</p> <p>Note: For MongoDB data sources, clicking Preview shows the filtered records, if one or more <i>where</i> clauses have been entered in the Where field. If no where clause has been entered, the preview displays all the records.</p> <p>Note: For Couchbase data sources, clicking Preview also displays the added <code>_id</code> field containing the key. If the record already has an <code>_id</code> field, then the added <code>_id</code> field overwrites the pre-existing one at the time of previewing the fields.</p>

Field Name	Description
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to pass to the next stage. For more information, see [Defining Fields in a NoSQL Database](#) on page 174

Defining Fields in a NoSQL Database

The **Fields** tab displays the field and its type as defined in the NoSQL DB/Schema file.

1. On the **Fields Tab**, click **Regenerate**.

This generates the aggregated data based on the first 50 records. The data is displayed in the following format: `Fieldname (datatype)`.

Note: If the schema file is browsed, the fields are generated using the schema file and the table or view is bypassed. To reset the schema file, click **Clear**.

Note: While reading data from a Couchbase DB, the key of each record is also read. This key is stored as a part of the record using an added `_id` field at the time of regenerating the fields, and is also included in the data sent to the next stage. If the record already has an `_id` field, then this would be overwritten with the added `_id` field at the time of regenerating the fields.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.

- You can also add additional fields that are not present in the table or schema file. Click **Add** to add a new field. To remove a field, click **Remove**.

Note: You can only add a new field under the List type.

- Click **OK**.

NoSQL DB Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for NoSQL DB.

- Open the dataflow in Enterprise Designer.
- If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
- Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
- Click **Add**. The **Define Dataflow Options** dialog box appears.
- Expand the NoSQLDB stage.
- The Dataflow options are exposed as described in the following table:

Database	Read	Write
Mongo DB	Connection	Connection
	Table	Table
Couchbase DB	Connection	Connection
	View	
	Design Document Name	

The selected NoSQL DB option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

- Enter a description of the option in the **Description** field.
- In the **Target** field, select the option **Selected stage(s)**.

9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Read from SAP

The Read from SAP stage reads data from an SAP database as input to a dataflow. It can read data from a single table or multiple tables. When reading data from multiple tables, the stage performs a join operation to determine which records to read into the dataflow.

Connecting to SAP

In order to read data from SAP into a dataflow using Read from SAP, you need to create a connection between Spectrum™ Technology Platform and your SAP system.

1. Open the SAP connection manager. You can do this in Enterprise Designer under **Tools > SAP Connection Management** or in the Read from SAP stage by clicking the **Manage** button next to the **Connection** field.
2. Click **Add**.
3. In the **Connection name** field, give this connection a name.
4. Complete the other fields with the information about the SAP server you want to connect to. See your SAP Basis administrator for the necessary information.

Important: The user ID and password must be for an SAP account with administrator privileges.

5. Click **Test** to verify the connection.
6. Click **OK**.

You have not created a connection that can be used by the Read from SAP stage to read data from SAP into a dataflow.

Reading Data from a Single SAP Table

The Read from SAP stage can be configured to read data from a single table in the SAP database or multiple tables. This procedure describes how to configure Read from SAP to read data from a single table.

1. In Enterprise Designer, drag Read from SAP onto the canvas.
2. Double-click the Read from SAP stage on the canvas.
3. In the **Connection** field, select the SAP server that contains the data you want to read into the dataflow. If there is no connection defined for the SAP server you need to create the connection by clicking **Manage**.
4. In the **Source type** field, choose **Single**.
5. Click **Select**.
6. Select the table you want to read into the dataflow then click **OK**.

Note: Only the first 200 tables are listed. Use the search feature to find tables not listed in the first 200. The search field only searches the values in the **Name** and **Label** columns.

7. To view the field names that will be used in the dataflow, check the box **Display technical name**.

Fields in SAP have a user-friendly name used for display purposes and a unique name that may be less readable. For example, a field may have a user-friendly name of "Distribution Channel" and a technical name of "DIS_CHANNEL". In order to ensure that the field name is valid in the dataflow, the technical name is used as the field name.

8. Check the box in the **Include** column for each field you want to read into the dataflow.
9. Click **OK**.
10. If you want to read only certain records, you can specify filter conditions on the **Filter** tab. In order for a record to be read into the dataflow it must meet all the conditions you define.
11. You can improve performance by specifying an appropriate fetch size on the **Runtime** tab.

Select this option to specify the number of records to read from the database table at a time. For example, if the **Fetch size** value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.

Setting an optimum fetch size can improve performance significantly.

Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see [Determining an Optimum Fetch Size](#) on page 294.

The default fetch size for Read from SAP is 10,000.

12. Click **OK**.

The Read from SAP stage is now configured to read data from a single table in the SAP database into the dataflow.

Reading Data from Multiple SAP Tables

The Read from SAP stage can be configured to read data from a single table in the SAP database or multiple tables. This procedure describes how to configure Read from SAP to read data from a multiple tables. To read data from multiple tables, you define a JOIN statement to combine data into a single stream.

1. In Enterprise Designer, drag Read from SAP onto the canvas.
2. Double-click the Read from SAP stage on the canvas.
3. In the **Connection** field, select the SAP server that contains the data you want to read into the dataflow. If there is no connection defined for the SAP server you need to create the connection by clicking **Manage**.
4. In the **Source type** field, choose **Multiple**.
5. Click **Add**.
6. Select the tables you want to read into the dataflow then click **OK**.

Note: Only the first 200 tables are listed. Use the search feature to find tables not listed in the first 200. The search field only searches the values in the **Name** and **Label** columns.

7. Select the first table in the list and click **Create Relationship**. This is the source table.
8. In the **Source key** field, select the column from the source table whose value will be used match records to records from the other table.
9. In the **Join type** field, select one of the following:

INNER JOIN	Returns only those records that have a match between the source and target tables.
LEFT JOIN	Returns all records from the source table even if there are no matches between the source and target tables. This option returns all records from the source table plus any records that match in the target table.
10. In the **Table** field, select the target table.
11. In the **Table key** field, select the column in the target table containing the data you want to compare to the data from the **Source key** field to determine if the record meets the join condition.
12. Click **OK**.
13. Click **Select Schema**.
14. Choose the fields that you want to read into the dataflow. To view the field names that will be used in the dataflow, check the **Display technical name** box.

Fields in SAP have a user-friendly name used for display purposes and a unique name that may be less readable. For example, a field may have a user-friendly name of "Distribution

Channel" and a technical name of "DIS_CHANNEL". In order to ensure that the field name is valid in the dataflow, the technical name is used as the field name.

15. Click **OK**.
16. If you want to read only certain records, you can specify filter conditions on the **Filter** tab. In order for a record to be read into the dataflow it must meet all the conditions you define.
17. You can improve performance by specifying an appropriate fetch size on the **Runtime** tab.

Select this option to specify the number of records to read from the database table at a time. For example, if the **Fetch size** value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.

Setting an optimum fetch size can improve performance significantly.

Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see [Determining an Optimum Fetch Size](#) on page 294.

The default fetch size for Read from SAP is 10,000.

The Read from SAP stage is now configured to read data from a multiple tables in the SAP database into the dataflow.

Filtering Records in Read from SAP

The filter settings in Read from SAP allow you to read a subset of records from an SAP table rather than all records in the table. To filter records, you specify the values that a record must contain in order for it to be read into the dataflow. If you do not specify any filter conditions, all records in the table are read into the dataflow. Using filter conditions is optional.

Note: If the Read from SAP stage is configured to read data from multiple SAP tables, the filter is applied after the JOIN operation is performed.

1. In the Read from SAP stage, click the **Filter** tab.
2. Click **Add**.
3. In the **Table name** field, select the table that contains the records you want to filter.
4. In the **Filter by** field, select the field that contains the data you want to use as the basis for filtering.
5. Choose one of the following operators:

Note: The operators available to you vary depending on the data type of the field on which you are filtering.

Operator	Description
Contains	Checks if the string contains the value specified.
Equals	Checks if the value in the field matches the value specified.
Not Equals	Checks if the value in the field does not match the value specified.
Greater Than	Checks if the field has a numeric value that is greater than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Greater Than Or Equals	Checks if the field has a numeric value that is greater than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Less Than	Checks if the field has a numeric value that is less than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Less Than Or Equals	Checks if the field has a numeric value that is less than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Null	Checks if the field is a null value.
Is Not Null	Checks if the field is not a null value.
Starts With	Checks if the field starts with the value specified.
Ends With	Checks if the field ends with the value specified.

6. Enter the value to which you want to compare the selected field's value.
7. Click **OK**.
8. Add additional filter conditions if needed.

Note: If you specify multiple filter conditions, all the filter conditions must be true in order for the record to be read into the dataflow. If any one of the conditions is not true, the record is not read into the dataflow.

Read from Spreadsheet

Read from Spreadsheet reads data from an Excel spreadsheet as input to a dataflow. It supports both .xls and .xlsx file formats.

File Properties Tab

The **File Properties** tab contains options for specifying the spreadsheet and data to read into the dataflow.

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.
File name	Specifies the path to the file. Click the ellipses button (...) to browse to the file you want. Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.
Data selection	Specifies how you want to select data from the spreadsheet to read into the dataflow. One of the following: <ul style="list-style-type: none"> Sheet Data Select this option to read in all the data from a sheet in the spreadsheet. Range Data Select this option to read in a subset of data from a sheet by specifying a range of cells to read. Named Range Select this option to read in a subset of data from a sheet by specifying a named range from the spreadsheet.
Sheet selection	If you choose Sheet Data or Range Data in the Data selection field, use this option to choose the sheet from which you want to read data into the dataflow.

Field Name	Description
Range	If you choose Range Data in the Data selection field, use this option to specify the cell that starts the range and the cell that ends the range.
Named range	If you choose Named Range in the Data selection field, use this option to specify the name of the range you want to read into the dataflow. Ranges are defined in the spreadsheet. If no ranges are listed it means that no ranges are defined in the spreadsheet.
Header row	<p>Check this box to specify a row that contains column headings. Column headings become the dataflow field names although you can change field names on the Fields tab. If you do not check this box, the dataflow fields are given generic default names Column1, Column2, and so on.</p> <p>The header row you specify is relative to the data selection. For example, if you choose Range Data in the Data selection field and the range begins on the fifth row, and you specify 1 as the header row, then the fifth row in the spreadsheet will be used as the header because the fifth row of the spreadsheet is the first row of the range.</p>
Data offset from header	If you specify a header row, this field specifies the first row that contains data, relative to the header. For example, if you specify 1, the first row below the header will be the first row of data to be read into the dataflow. If you specify 2, the second row below the header will be the first row read into the dataflow, and so on.
First data row	If you do not specify a header row, this field specifies which row within the data selection contains the first row of data to read into the dataflow. The row you specify is relative to the data selection. For example, if you choose Range Data in the Data selection field and the range begins on the fifth row, and you specify 1 as the first data row, then the first row of data to be read into the dataflow will be the fifth row.
Ignore empty rows	<p>Select this option if you want empty rows to be excluded from the dataflow. If you do not select this option, empty rows in the spreadsheet will result in empty records in the dataflow.</p> <p>Note: This option does not affect the data shown in the preview. Empty rows are always shown in the preview even if this option is selected.</p>

Fields Tab

The **Fields** tab contains options for mapping the data from the spreadsheet to fields in the dataflow.

Option	Description
Regenerate	Click this button to populate the fields tab with the fields in the input file defined on the File Properties tab.
Detect Type	Click this button to automatically determine the data type for all the fields. You can manually change a field's data type by selecting the field and clicking Modify .
Modify	Select a field then click this button to modify the field name or data type.

Read from Variable Format File

Read from Variable Format File reads data from a file containing records of varying layout. Each record is read in as a list field. You can specify the tag that indicates the parent record type, and all other record types will become list fields under the parent.

Variable format files have these characteristics:

- Records in the file may have different fields, and different numbers of fields.
- Each record must contain a tag (usually a number) identifying the type of record.
- Hierarchical relationships are supported.

Example of a Variable Format File

This example shows a variable format file containing information about checking account activity for two customers, Joe Smith and Anne Johnson. In this example, the file is a delimited file that uses a comma as the field delimiter.

```
001 Joe,Smith,M,100 Main St,555-234-1290
100 CHK12904567,12/2/2007,6/1/2012,CHK
200 1000567,1/5/2012,Fashion Shoes,323.12
001 Anne,Johnson,F,1202 Lake St,555-222-4932
100 CHK238193875,1/21/2001,4/12/2012,CHK
200 1000232,3/5/2012,Blue Goose Grocery,132.11
200 1000232,3/8/2012,Trailway Bikes,540.00
```

The first field in each record contains the tag which identifies the type of record and therefore the record's format:

- 001: Customer record

- 100: Account record
- 200: Account transaction record

For delimited files it is common for the tag value (001, 100, 200) to be in a fixed number of bytes at the start of the record as shown in the above example.

Each record has its own format:

- 001: FirstName,LastName,Gender,Address,PhoneNumber
- 100: AccountID,DateOpened,ExpirationDate,TypeOfAccount
- 200: TransactionID,DateOfTransaction,Vendor,Amount

Record format 100 (account record) is a child of the previous 001 record, and record format 200 (account transaction record) is a child of the previous record 100 (account record). So in the example file, Joe Smith's account CHK12904567 had a transaction on 1/5/2012 in the amount of 323.12 at Fashion Shoes. Likewise, Anne Johnson's account CHK238193875 had two transactions, one on 3/5/2012 at Blue Goose Grocery and one on 3/8/2012 at Trailway Bikes.

File Properties Tab

Option Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.
File name	<p>Specifies the path to the file. Click the ellipses button (...) to go to the file you want.</p> <p>You can read multiple files by using a wild card character to read data from multiple files in the directory. The wild card characters * and ? are supported. For example, you could specify *.CSV to read in all files with a .CSV extension located in the directory. In order to successfully read multiple files, each file must have the same layout (the same fields in the same positions). Any record that does not match the layout specified on the Fields tab will be treated as a malformed record.</p> <p>While reading a file from an HDFS file server, the compression formats supported are:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: The extension of the file indicates the compression format to be used to decompress the file.</p> <p>Attention: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>

Option Name	Description
Record type	<p data-bbox="553 331 1089 363">The format of the records in the file. Select one of:</p> <p data-bbox="553 380 1425 474">Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.</p> <p data-bbox="553 491 1425 585">Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.</p> <p data-bbox="553 602 1425 730">Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.</p>

Option Name	Description
Character encoding	<p>The text file's encoding. Select one of these:</p> <p>UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>US-ASCII A character encoding based on the order of the English alphabet.</p> <p>UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first).</p> <p>UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first).</p> <p>ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1.</p> <p>ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.</p> <p>ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5.</p> <p>CP850 An ASCII code page used to write Western European languages.</p> <p>CP500 An EBCDIC code page used to write Western European languages.</p> <p>Shift_JIS A character encoding for the Japanese language.</p> <p>MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.</p> <p>CP1047 An EBCDIC code page with the full Latin-1 character set.</p>
Record length	For fixed width files, specifies the exact number of characters in each record.

Option Name	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre data-bbox="553 430 1421 493">7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul data-bbox="553 556 690 751" style="list-style-type: none">• Space• Tab• Comma• Period• Semicolon• Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Tag separator	<p>Specifies the character placed after the tag field to demarcate the identifying field for each record in a delimited file. A tag separator must be a single character.</p> <p>By default, these characters are available to be selected as tag separators:</p> <ul data-bbox="553 1018 690 1213" style="list-style-type: none">• Space• Tab• Comma• Period• Semicolon• Pipe <p>If the file uses a different character as a tag separator, click the ellipses button to add and select a custom tag separator.</p> <p>Note: By default, the Record separator character is the same as the selected Field separator character. To enable this field and select a different character, uncheck the Same as Field separator checkbox.</p>
Same as Field separator	<p>Indicates if the tag separator is the same as the field separator. Uncheck this to select a different character as the tag separator.</p> <p>Note: By default, this checkbox is checked and the Tag separator field is disabled.</p>

Option Name	Description
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <p>Unix (U+000A) A line feed character separates the records. This is the standard record separator for Unix systems.</p> <p>Macintosh (U+000D) A carriage return character separates the records. This is the standard record separator for Macintosh systems.</p> <p>Windows (U+000D U+000A) A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</p> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>
Root tag name	<p>The tag to use for records that are a parent of other record types. For example if you have three record types 001, 100, and 200, and record types 100 and 200 are children of record type 001, then 001 is the root tag.</p>
Use fixed-width tags	<p>Specifies whether to allocate a fixed amount of space at the beginning of each record in which to place the record tag. For example, the following shows a file with the tags 001, 100, and 200 in a fixed-width field:</p> <pre>001 Joe, Smith, M, 100 Main St, 555-234-1290 100 CHK12904567, 12/2/2007, 6/1/2012, CHK 200 1000567, 1/5/2012, Mike's Shoes, 323.12</pre>
Tag start position	<p>If you check the Use fixed-width tags box, this option specifies the position in each record where the tag begins. For example, if the tag begins in the fourth character in the record, you would specify 4.</p>

Option Name	Description
Tag width	<p>If you check the Use fixed-width tags box, this option specifies the number of spaces to allocate for tags starting from the position specified in the Tag start position field. For example, if you specify 3 in the Tag start position field and you specify 7 in the Tag width field, then positions 4 through 10 would be considered the record tag. The value you specify must be large enough to include all the characters of the longest tag name.</p> <p>The value in the Tag width field is automatically increased if you lengthen the tag name in the Root tag name field.</p> <p>The maximum tag width is 1024.</p>
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum™ Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Treat records with fewer fields than defined as malformed	<p>If you enable this option, child records that contain fewer fields than a complete record are considered malformed. When a malformed record is encountered, processing advances to the next root tag, ignoring all child tags in between. An exception is written to the log containing information about the malformed child records along with a line number.</p> <p>Note: Records are always considered malformed in the following situations, regardless of whether you enable this option.</p> <ul style="list-style-type: none"> • The tag is unknown • The line is empty • There is a tag with no data • A record with a tag that is a child of another tag appears immediately after a record with a root tag

Fields Tab

The **Fields** tab specifies the characteristics of each field read in from the file.

[Defining Fields in Delimited Variable Format Files](#) on page 190

[Defining Fields in a Line Sequential or Fixed Width Variable Format File](#) on page 193

Runtime Tab

Field Name	Description
File name	Displays the file name selected in the first tab.
Starting record	If you want to skip records at the beginning of the file when reading records into the dataflow, specify the first record you want to read. For example, if you want to skip the first 50 records, in a file, specify 51. The 51st record will be the first record read into the dataflow.
All records	Select this option if you want to read all records starting from the record specified in the Starting record field to the end of the file.
Max records	Select this option if you want to only read in a certain number of records starting from the record specified in the Starting record field. For example, if you want to read the first 100 records, select this option and enter 100.

Defining Fields in Delimited Variable Format Files

This procedure describes how to define fields in the Read from Variable Format File stage for delimited files.

1. In the Read from Variable Format File stage, click the **Fields** tab.
2. Click **Regenerate**.

A list of all the fields for each record type is displayed. For each field the following information is displayed:

Parent	The tag from the input file indicating the record type in which the field appears. If the tag begins with a number, the tag is prefixed with "NumericTag_". For example, a tag named 100 would become NumericTag_100. The prefix is necessary because dataflow field names cannot begin with a number.
Field	The name that will be used in the dataflow for the field. By default, fields are given names in the format <Tag Name>_<Column n>. For example, the first field of record type Owner would be Owner_Column1, the second would be Owner_Column2, and so on.
Type	The field's data type.

Note: The first 50 records are used to generate the fields list. The input file must contain at least two root tags in order to generate a fields list.

3. If you want to modify the parent/child relationships between the tags:
 - a) Click **Modify Tag Hierarchy**.
 - b) Click and drag the tags to define the tag hierarchy you want.
 - c) Click **OK**.
4. If you want to modify the a field's name or data type, select the field and click **Modify**.
5. In the **Name** field, choose the field you want to add or type the name of the field.

Typically you will want to replace the default names with meaningful names to represent the data in the field. For example, consider this input data:

```
001 Joe,Smith,M,100 Main St,555-234-1290
```

This record has a parent tag of 001 and would have these fields created by default:

```
NumericTag_001_Column1: Joe
NumericTag_001_Column2: Smith
NumericTag_001_Column3: M
NumericTag_001_Column4: 100 Main St
NumericTag_001_Column5: 555-234-1290
```

You would probably want to rename the fields so that the names describe the data. For example:

```
FirstName: Joe
LastName: Smith
Gender: M
AddressLine1: 100 Main St
PhoneNumber: 555-234-1290
```

Note: You cannot rename list fields. List fields, which contain all the fields for a given record type, always use the tag name from the input file as the field name.

6. To change a field's data type, select the data type you want in the **Type** field.

The following data types are available:

- bigdecimal** A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.
- boolean** A logical type with two values: true and false.
- bytearray** An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre> <Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names> </pre> <p>It is important to note that the Spectrum™ Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.</p>
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

- If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Management Console or Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
- b) In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

8. Click **OK**.

Defining Fields in a Line Sequential or Fixed Width Variable Format File

This procedure describes how to define fields in the Read from Variable Format File stage for line sequential or fixed width files.

1. In the Read from Variable Format File stage, click the **Fields** tab.
2. Click **Get Tags**.

A list of all the fields for each record type is displayed. For each field the following information is displayed:

Parent	The tag from the input file indicating the record type in which the field appears. If the tag begins with a number, the tag is prefixed with "NumericTag_". For example, a tag named 100 would become NumericTag_100. The prefix is necessary because dataflow field names cannot begin with a number.
Field	The name that will be used in the dataflow for the field. By default, fields are given names in the format <Tag Name>_<Column n>. For example, the first field of record type Owner would be Owner_Column1, the second would be Owner_Column2, and so on.
Type	The field's data type.

Note: The first 50 records are used to generate the fields list. The input file must contain at least two root tags in order to generate a fields list.

3. In the **Filter** field, select the tag for the record type whose fields you want to define then click **Add**.

Note: The filter does not have any impact on which fields are read into the dataflow. It only filters the list of fields to make it easier to browse.

4. In the **Name** field, choose the field you want to add or type the name of the field.
5. In the **Type** field, you can leave the data type as `string` if you do not intend to perform any mathematical or date time operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum™ Technology Platform supports these data types:

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field

Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum™ Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.

- long** A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
- string** A sequence of characters.
- time** A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

6. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Management Console or Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
- b) In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify

your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

- In the **Start position** field, enter the position of the first character of the field, and in the **Length** field enter the number of characters in the field.

For example, if the field starts at the tenth character of the record and is five characters long, you would specify a starting position of 10 and a length of 5.

- Click **Add**.
- Repeat this process to add additional fields to the record type, or click **Close** if you are done adding fields.

Flattening Variable Format Data

Variable format file data often contains records that have a hierarchical relationship, with one record type being a parent to other record types. Since many stages require data to be in a flat format, so you may have to flatten the data in order to make the data usable by downstream stages. For example, consider this input data:

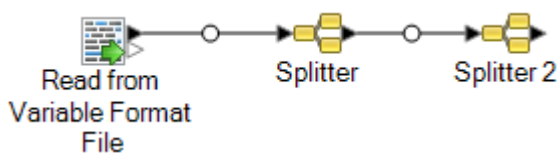
```
001   Joe,Smith,M,100 Main St,555-234-1290
100   CHK12904567,12/2/2007,6/1/2012,CHK
200   1000567,1/5/2012,Fashion Shoes,323.12
001   Anne,Johnson,F,1202 Lake St,555-222-4932
100   CHK238193875,1/21/2001,4/12/2012,CHK
200   1000232,3/5/2012,Blue Goose Grocery,132.11
200   1000232,3/8/2012,Trailway Bikes,540.00
```

You may want to flatten the records so that you have one record per transaction. In the above example, that would mean taking the transaction records (records with the tag 200) and flattening them to include the account owner information (records with the tag 001) and the account details (records with the tag 100).

The following procedure describes how to use Splitter stages to flatten records.

- Add a Read from Variable Format File stage to your data flow and configure the stage. For more information, see [Read from Variable Format File](#) on page 183.
- Add a Splitter stage and connect it to Read from Variable Format File.
- Add additional Splitter stages as needed so that you have one splitter stage for each child record type in your input data.
- Connect all the Splitter stages.

You should now have a data flow that looks like this:



5. Double-click the first Splitter stage to open the stage options.
6. In the **Split at** field, select one of the child record types.
7. Click **OK**.
8. Configure each additional Splitter stage, selecting a different child record type in each Splitter's **Split at** field.

Read From XML

The Read from XML stage reads an XML file into a job or subflow. It defines the file's path and data format, including XML schema and data element details.

Simple XML elements are converted to flat fields and passed on to the next stage. Simple XML data consists of records made up of XML elements that contain only data and no child elements. For example, this is a simple XML data file:

```
<customers>
  <customer>
    <name>Sam</name>
    <gender>M</gender>
    <age>43</age>
    <country>United States</country>
  </customer>
  <customer>
    <name>Jeff</name>
    <gender>M</gender>
    <age>32</age>
    <country>Canada</country>
  </customer>
  <customer>
    <name>Mary</name>
    <gender>F</gender>
    <age>61</age>
    <country>Australia</country>
  </customer>
</customers>
```

Notice that in this example each record contains simple XML elements such as `<name>`, `<gender>`, `<age>`, and `<country>`. None of the elements contain child elements.

The Read from XML stage automatically flattens simple data like this because most stages require data to be in a flat format. If you want to preserve the hierarchical structure, use an Aggregator stage after Read from XML to convert the data to hierarchical data.

Complex XML elements remain in hierarchical format and are passed on as a list field. Since many stages require data to be in a flat format, so you may have to flatten the complex XML elements in order to make the data usable by downstream stages. For more information, see [Flattening Complex XML Elements](#) on page 202.

Note: Read From XML does not support the XML types `xs:anyType` and `xs:anySimpleType`.

File Properties Tab

Table 4: File Properties Tab

Option Name	Description
Schema file	<p>Specifies the path to an XSD schema file. Click the ellipses button (...) to browse to the file you want. Note that the schema file must be on the server in order for the data file to be validated against the schema. If the schema file is not on the server, validation is disabled.</p> <p>Alternatively, you can specify an XML file instead of an XSD file. If you specify an XML file the schema will be inferred based on the structure of the XML file. Using an XML file instead of an XSD file has the following limitations:</p> <ul style="list-style-type: none"> • The XML file cannot be larger than 1 MB. If the XML file is more than 1 MB in size, try removing some of the data while maintaining the structure of the XML. • The data file will not be validated against the inferred schema. <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Data file	<p>Specifies the path to the XML data file. Click the ellipses button (...) to browse to the file you want.</p> <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Preview	<p>Displays a preview of the schema or XML file. When you specify an XSD file, the tree structure reflects the selected XSD. Once you specify both a schema file and a data file, you can click on the schema elements in bold to see a preview of the data that the element contains.</p>

*Fields Tab***Table 5: Fields Tab**

Option Name	Description
Filter	Filters the list of elements and attributes to make it easier to browse. The filter does not have any impact on which fields are included in the output. It only filters the list of elements and attributes to make it easier to browse.
XPath	The XPath column displays the XPath expression for the element or attribute. It is displayed for information purposes only. For more information on XPath, see www.w3schools.com/xpath/ .
Field	The name that will be used in the dataflow for the element or attribute. To change the field name, double-click and type the field name you want.

Option Name	Description
-------------	-------------

Type

Option Name

Description

The data type to use for the field.

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

date A data type that contains a month, day, and year. Dates must be in the format *yyyy-MM-dd*. For example, 2012-01-30.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. Datetime must be in the format *yyyy-MM-dd'T'HH:mm:ss*. For example, 2012-01-30T06:15:30

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52})\times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23})\times 2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum™ Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9223372036854775808) and $2^{63}-1$ (9223372036854775807).

string A sequence of characters.

time A data type that contains the time of day. Time must be in the format

Option Name	Description
	<i>HH:mm:ss</i> . For example, 21:15:59.
Include	Specifies whether to make this field available in the dataflow or to exclude it.

Example: Simple XML File

In this example, you want to read the following file into a dataflow:

```
<addresses>
  <address>
    <addressline1>One Global View</addressline1>
    <city>Troy</city>
    <state>NY</state>
    <postalcode>12128</postalcode>
  </address>
  <address>
    <addressline1>1825B Kramer Lane</addressline1>
    <city>Austin</city>
    <state>TX</state>
    <postalcode>78758</postalcode>
  </address>
</addresses>
```

In this example, you could choose to include the `<addressline1>`, `<city>`, `<state>`, and `<postalcode>`. This would result in one record being created for each `<address>` element because `<address>` is the common parent element for `<addressline1>`, `<city>`, `<state>`, and `<postalcode>`.

Flattening Complex XML Elements

Most stages in a dataflow require data to be in a flat format. This means that when you read hierarchical data from an XML file into a dataflow, you will have to flatten it if the data contains complex XML elements. A complex XML element is an element that contains other elements or attributes. For example, in the following data file the `<address>` element and the `<account>` element are complex XML elements:

```
<customers>
  <customer>
    <name>Sam</name>
    <gender>M</gender>
```

```

    <age>43</age>
    <country>United States</country>
    <address>
      <addressline1>1253 Summer St.</addressline1>
      <city>Boston</city>
      <stateprovince>MA</stateprovince>
      <postalcode>02110</postalcode>
    </address>
    <account>
      <type>Savings</type>
      <number>019922</number>
    </account>
  </customer>
  <customer>
    <name>Jeff</name>
    <gender>M</gender>
    <age>32</age>
    <country>Canada</country>
    <address>
      <addressline1>26 Wellington St.</addressline1>
      <city>Toronto</city>
      <stateprovince>ON</stateprovince>
      <postalcode>M5E 1S2</postalcode>
    </address>
    <account>
      <type>Checking</type>
      <number>238832</number>
    </account>
  </customer>
  <customer>
    <name>Mary</name>
    <gender>F</gender>
    <age>61</age>
    <country>Australia</country>
    <address>
      <addressline1>Level 7, 1 Elizabeth Plaza</addressline1>
      <city>North Sydney</city>
      <stateprovince>NSW</stateprovince>
      <postalcode>2060</postalcode>
    </address>
    <account>
      <type>Savings</type>
      <number>839938</number>
    </account>
  </customer>
</customers>

```

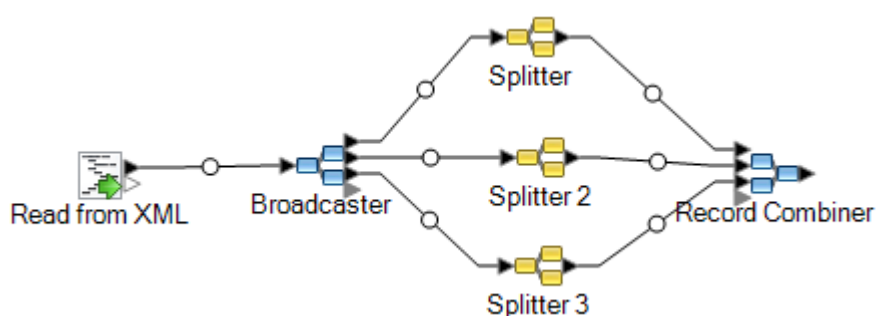
The following procedure describes how to use Splitter stages to flatten XML data containing multiple complex XML elements.

Note: If your data contains a single complex XML element, you can use a single Splitter stage to flatten the data by simply connecting the Read from XML stage to the Splitter stage.

You do not need to use the Broadcaster and Record Combiner stages as described in this procedure for data files containing a single complex XML element.

1. Add a Read from XML stage to your data flow and configure the stage. For more information, see [Read From XML](#) on page 197.
2. Add a Broadcaster stage and connect Read from XML to it.
3. Add a Splitter stage for each complex XML element in your data.
4. Connect the Broadcaster stage to each Splitter.
5. Add a Record Combiner stage and connect each Splitter to it.

You should now have a data flow that looks like this:



6. Double-click the first Splitter stage to open the stage options.
7. In the **Split at** field, select one of the complex fields. In the example data file above, this could be the address field.
8. Click **OK**.
9. Configure each additional Splitter stage, selecting a different complex XML element in each Splitter's **Split at** field.

The data flow is now configured to take XML input containing records with complex XML elements and flatten the data. The resulting records from Record Combiner can be sent to any stage that requires flat data. For example, you could attached the Record Combiner stage to a Validate Address stage for address validation.

SQL Command

SQL Command executes one or more SQL commands for each record in the dataflow. You can use SQL Command to:

- Execute complex INSERT/UPDATE statements, such as statements that have subqueries/joins with other tables.
- Update tables after inserting/updating data to maintain referential integrity.
- Update or delete a record in a database before a replacement record is loaded.

- Update multiple tables in a single transaction.

You can execute additional SQL commands before and after executing the main SQL commands, and you can invoke stored procedures.

Note: Stored procedures invoked from SQL Command must not use OUT parameters.

Note: Significant performance improvements can be achieved by using multiple runtime instances of SQL Command. To specify multiple runtime instances, click the **Runtime** button.

General

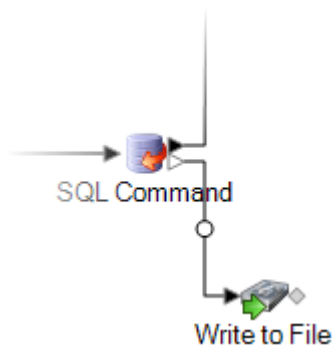
The **General** tab is where you specify dynamic SQL statements that you want to execute once for each record. The following table lists the options available on the **General** tab.

Option	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
SQL statements	<p>Enter the SQL statements you want to execute for each record in the dataflow. As you begin to type, an auto-complete pop-up window will display the valid SQL commands. Separate multiple SQL statements with a semicolon (;).</p> <p>To specify a value from a dataflow field, use this syntax:</p> <pre> \${<field name>} </pre> <p>Where <field name> is the name of a field in the dataflow.</p> <p>For example:</p> <pre> UPDATE MyDatabase.dbo.customer SET name=\${Name} WHERE id=\${ID}; </pre> <p>In this example <code>\${Name}</code> will be replaced with the value from the dataflow's Name field and <code>\${ID}</code> will be replaced with the value from the dataflow's ID field.</p> <p>Note: Queries must use the fully-qualified name. For example, <code>MyDatabase.dbo.customer</code>.</p>

Option	Description
Transaction processing	<p>Specifies whether to process records in batches or to process all records at the same time. One of the following:</p> <p>Batch size Groups records into batches of the size you specify and processes one batch at a time.</p> <p>Entire Run Creates one large batch for all records and processes all transactions at the same time.</p>
Error processing	<p>Specifies what to do if an error is encountered while executing the SQL commands. One of the following:</p> <p>Do not terminate the dataflow on error The dataflow continues to run if the database returns an error while executing the SQL commands.</p> <p>Terminate the dataflow after encountering this many errors The dataflow will stop running after the database returns the specified number of errors.</p>

Note: If there is a syntax error in the SQL, the dataflow will always terminate regardless of which setting you choose here.

In addition, you can optionally write error records to a sink by connecting the SQL Command error port to the type of sink you want. The error port is the white triangle on the right side of the stage icon in the dataflow. For example, to write error records to a flat file, you would connect the SQL Command error port to a Write to File stage, as shown here:



Pre/Post SQL

The **Pre/Post SQL** tab is where you specify SQL statements that you want to execute once per dataflow run, as opposed to once per record as is the case with the SQL you specify on the **General** tab. The following table lists the options available on the **Pre/Post SQL** tab.

Option	Description
Pre-SQL	<p>Type one or more SQL statements that you want to execute before the records coming into the stage are processed. The SQL statements you enter here are executed once per run after the dataflow starts running but before the SQL Command stage processes the first records.</p> <p>An example use of pre-SQL would be to create a table for the records that will be processed.</p>
Autocommit pre-SQL	<p>Check this box to commit the pre-SQL statements before executing the SQL statements on the General tab.</p> <p>If you do not check this box, the pre-SQL statements will be committed in the same transaction as the SQL statements on the General tab.</p> <p>Note: If you check neither the Autocommit pre-SQL nor the Autocommit post-SQL boxes, then all SQL statements for the stage are committed in one transaction.</p>
Post-SQL	<p>Type one or more SQL statements that you want to execute after all the records are processed. The SQL statements you enter here are executed once per run after the SQL Command stage is finished but before the dataflow finishes.</p> <p>An example use of pre-SQL would be to build an index after processing the records.</p>
Autocommit post-SQL	<p>Check this box to commit the post-SQL statements in their own transaction after the SQL commands on the General tab are committed.</p> <p>If you do not check this box, the post-SQL statements will be committed in the same transaction as the SQL statements on the General tab.</p> <p>Note: If you check neither the Autocommit pre-SQL nor the Autocommit post-SQL boxes, then all SQL statements for the stage are committed in one transaction.</p>

Runtime Tab

The **Runtime** tab displays **Stage Options** and gives you the flexibility of defining default values for the stage options.

Field Name	Description
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Specifying SQL Command at Runtime

This procedure describes how to configure a dataflow to support runtime options for SQL Command and also how to specify the job executor arguments to do this.

1. Open the dataflow in Enterprise Designer.
2. If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
3. Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
4. Click **Add**. The **Define Dataflow Options** dialog box appears.
5. Expand the SQL Command stage.
6. Select a SQL Command option. It can be **PreSqlCommand**, **SqlCommand**, or **PostSqlCommand**.

PreSqlCommand SQL statements that you want to execute before the records coming into the stage are processed. These SQL statements are executed once per run after the dataflow starts running but before the SQL Command stage processes the first record.

An example use of pre-SQL would be to create a table for the records that will be processed.

SqlCommand SQL statements you want to execute for each record in the dataflow.

PostSqlCommand SQL statements that you want to execute after all the records are processed. These SQL statements are executed once per run after the SQL Command stage is finished but before the dataflow finishes.

An example use of post-SQL would be to build an index after processing the records.

The selected SQL Command option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.
15. Create a text file containing the SQL statement you want to use at runtime.

The text file may look like this:

```
SqlCommand = UPDATE CustomersSET
ContactName='Alfred Schmidt'
City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
```

In this example, SqlCommand is one of the SQL Command stage's option names.

16. Use the -o argument when running a job executor from command line.

```
java -jar jobexecutor.jar -h "noipa019sh-11" -u "admin" -p "admin" -s
"8080" -o "options.txt" -j "FetchOracleData" -w
```

The filename (options.txt) specifies a name of the text file that you created in step 14.

For more information, see [Running A Job from the Command Line](#) on page 209

Running A Job from the Command Line

Before you can run a job from the command line, it must be exposed. To expose a job, open the job in Enterprise Designer and select **File > Expose/Unexpose and Save**.

To run a job from the command line, you must install the job executor utility on the system where you want to run the job. The Job Executor is available from the Spectrum™ Technology Platform Welcome page on the Spectrum™ Technology Platform server (for example, http://myserver:8080).

Usage

```
java -jar jobexecutor.jar -u UserID -p Password -j Job [Optional Arguments]
```

Required	Argument	Description
No	-?	Prints usage information.
No	-d <i>delimiter</i>	Sets instance/status delimiter. This appears in synchronous output only.
No	-e	Use a secure HTTPS connection for communication with the Spectrum™ Technology Platform server.
No	-f <i>property file</i>	Specifies a path to a job property file. A job property file contains job executor arguments. For more information on job property files, see Using a Job Property File .
No	-h <i>host name</i>	Specifies the name or IP address of the Spectrum™ Technology Platform server.
No	-i <i>poll interval</i>	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	-j <i>job name</i>	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.
No	-n <i>email list</i>	Specifies a comma-separated list of additional email addresses for configured job notifications.
No	-o <i>property file</i>	Specifies a path to a dataflow options property file. Use a dataflow options property file to set options for stages in the dataflow. In order to set dataflow options using a property file, you must configure the dataflow to expose stage options at runtime. For more information, see Adding Dataflow Runtime Options . For example, a dataflow options properties file for a dataflow that contains an Assign GeoTAX Info stage may look like this:
		<pre>OutputCasing=U UseStreetLevelMatching=N TaxKey=T Database.GTX=gsl</pre>
Yes	-p <i>password</i>	The password of the user.

Required Argument	Description
No <code>-r</code>	<p>Specify this argument to return a detailed report about the job. This option only works if you also specify <code>-w</code>. The report contains the following information:</p> <ul style="list-style-type: none"> • Position 1—Name of job • Position 2—Job process ID • Position 3—Status • Position 4—Start Date/Time (MM/DD/YYYY HH:MM:SS) • Position 5—End Date/Time (MM/DD/YYYY HH:MM:SS) • Position 6—Number of successful records • Position 7—Number of failed records • Position 8—Number of malformed records • Position 9—Currently unused <p>For example:</p> <pre>MySimpleJob 4 succeeded 04/09/2010 14:50:47 04/09/2010 14:50:47 100 0 0 </pre> <p>The information is delimited using the delimiter specified in the <code>-d</code> argument.</p>
No <code>-s port</code>	<p>The socket (port) on which the Spectrum™ Technology Platform server is running. The default is 8080.</p>
No <code>-t timeout</code>	<p>Sets the timeout (in seconds) for synchronous mode. The default is 3600. The maximum is 2147483. This is a global, aggregate timeout and represents the maximum time to wait for all spawned jobs to complete.</p>
Yes <code>-u user name</code>	<p>The login name of the user.</p>
No <code>-v</code>	<p>Return verbose output.</p>
No <code>-w</code>	<p>Runs job executor in synchronous mode. This means that job executor remains running until the job completes.</p> <p>If you do not specify <code>-w</code>, job executor exits after starting the job, unless the job reads from or writes to files on the server. In this case,</p>

Required Argument	Description
	job executor will run until all local files are processed, then exit.
No <i>StageName=Protocol:FileName</i>	Overrides the input or output file specified in Read from File or Write to File. For more information, see Overriding Job File Locations .
No <i>StageName:schema=Protocol:SchemaFile</i>	Overrides the file layout definition specified in Read from File or Write to File with one defined in a schema file. For more information, see Overriding the File Format at the Command Line .

Example Use of Job Executor

The following example shows command line invocation and output:

```
D:\spectrum\job-executor>java -jar jobexecutor.jar -u user123
-p "mypassword" -j validateAddressJob1 -h spectrum.example.com
-s 8888 -w -d "%" -i 1 -t 9999
```

```
validateAddressJob1%105%succeeded
```

In this example, the output indicates that the job named 'validateAddressJob1' ran (with identifier 105) with no errors. Other possible results include "failed" or "running."

Executing SQL Commands Before or After a Dataflow

The **Execute SQL** activity performs operations on database at any point during a process flow. This activity allows you to run the SQL statements both before and after the execution of Spectrum™ Technology Platform dataflow or an external program. For example, the **Execute SQL** activity can be used to delete indexes before the execution of a Spectrum™ Technology Platform dataflow and to create indexes again after the execution of the dataflow. To execute SQL statements using **Execute SQL** activity, you must create a process flow.

Note: Please refer to *Dataflow Designer Guide* for instructions on how to create and schedule a process flow.

1. Drag the **Execute SQL** activity to the canvas.
2. Double click the **Execute SQL** activity.
3. Select a database connection you want to use.

If you need to make a new database connection, or modify or delete an existing database connection, click **Manage**.

If you are adding or modifying a database connection, complete these fields:

Connection name	Enter a name for the connection. The name can be anything you choose.
Database driver	Select the appropriate database type.
Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.

- Write the SQL statement in the **SQL statement(s)** box.

By default, the **Terminate flow on error** option is checked which means that the process flow will be terminated if an exception occurs. If the option **Terminate flow on error** is unchecked and an exception occurs, the process flow will not stop and the exception will be logged in the server logs.

- Add the action you want a process flow to perform.

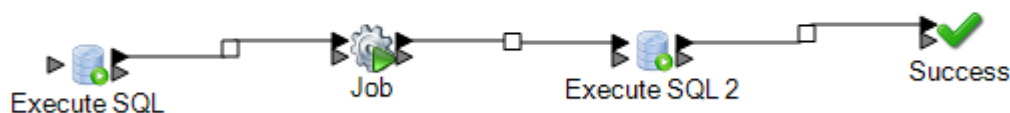
You can add a job by dragging a job's icon to the canvas, or add an external program by dragging a Run Program icon onto the canvas.

- Connect the two activities.

- Add additional **Execute SQL** activity as needed.

Refer step 2 to step 5 for performing actions on **Execute SQL**.

- When you have added all the jobs, Run Program and Execute SQL activities you want to execute in the process flow, drag a Success activity onto the canvas and connect it to the last activity in the process



flow.

- Run the Process flow.

Transposer

Transposer converts columns to rows. Transposing data is the opposite of pivoting data using the Group Statistics stage, which transforms row data into columns.

To understand Transposer, consider the following example. A table contains four quarters of sales data and you want to add all the revenue generated and to analyze the growth achieved in first three quarters. To accomplish this, use Transposer to create a column containing all the revenue of three

transposed quarters. Using Transposer to add all the revenues generated in different columns into a column potentially improves performance instead of adding them in different columns.

The following table explains the options in the Transposer dialog box.

Option	Description
Transposed fields header	Type a header name for the column that will contain those columns which are to be transposed. This new column is automatically added to the dataflow.
Transposed values header	Type a header name for the column that will contain the transposed column values. This new column is automatically added to the dataflow.
Retain transposed fields	Check this option to retain all the transposed fields as columns in the output.
Field Name	Displays all the column headers of input file.
Type	Displays the data type of the respective fields (column headers). The columns to be transposed should have compatible data type in the input source file. Below is the compatibility matrix. The tick marked grids correspond to the compatible data types.

	Integer	Long	String	Date/Time	Double	Big Decimal	Time	Date
Integer	✓	✓	✓		✓	✓		
Long	✓	✓	✓		✓	✓		
String	✓	✓	✓	✓	✓	✓	✓	✓
Datetime			✓	✓				
Double	✓	✓	✓		✓			
Bigdecimal	✓	✓	✓			✓		
Time			✓				✓	
Date			✓					✓

Option	Description
Transposed	Check the box next to each field that you want to convert to a column. In order to prevent a column from getting transposed and retain it in the output, clear the check box.

Example Use of Transposer

The following input data contains four quarters of sales by store. Note that Q1, Q2, Q3, and Q4 represent four quarters of sales (in millions).

Store (US)	Q1	Q2	Q3	Q4
New York	100.00	200.10	300.00	400.00
California	250.10	450.00	550.00	650.00
Illinois	150.00	250.10	350.00	450.00

The cases mentioned below illustrate the behavior of Transposer using the options provided in the stage. Note that Quarter is the column name for Transposed fields header and Revenue is the column name for Transposed fields values.

Case 1

Suppose you want columns Q1, Q2, and Q3 to be transposed and Q4 to be retained in the output. To do this, check the box under the **Transposed** header next to each column which is to be transposed. You will now see Q1, Q2, and Q3 as rows whereas Q4 will be retained as a column in the output.

Store (US)	Quarter	Revenue	Q4
New York	Q1	100.00	400.00
New York	Q2	200.10	400.00
New York	Q3	300.00	400.00
California	Q1	250.10	650.00
California	Q2	450.00	650.00

Store (US)	Quarter	Revenue	Q4
California	Q3	550.00	650.00
Illinois	Q1	150.00	450.00
Illinois	Q2	250.10	450.00
Illinois	Q3	350.00	450.00

Case 2

Suppose you want columns Q1 and Q2 to be transposed and Q3 and Q4 to be retained in the output. In addition, you also want to retain all the transposed fields (Q1 and Q2) as columns in the output. To do this, check the option **Retain transposed fields** and the box under the **Transposed** header next to each column to be transposed. You will now see Q1 and Q2 as rows whereas Q3 and Q4 will be retained as columns in the output along with Q1 and Q2.

Store (US)	Quarter	Revenue	Q1	Q2	Q3	Q4
New York	Q1	100.00	100.00	200.10	300.00	400.00
New York	Q2	200.10	100.00	200.10	300.00	400.00
California	Q1	250.10	250.10	450.00	550.00	650.00
California	Q2	450.00	250.10	450.00	550.00	650.00
Illinois	Q1	150.00	150.00	250.10	350.00	450.00
Illinois	Q2	250.10	150.00	250.10	350.00	450.00

Unique ID Generator

The Unique ID Generator stage creates a unique key that identifies a specific record. A unique ID is crucial for data warehouse initiatives in which transactions may not carry all name and address data, but must be attributed to the same record/contact. A unique ID may be implemented at the individual, household, business, and/or premises level. Unique ID Generator provides a variety of algorithms to create unique IDs.

The unique ID is based on either a sequential number or date and time stamp. In addition, you can optionally use a variety of algorithms to generate data to be appended to the ID, thereby increasing the likelihood that the ID will be unique. The sequential number or date and time stamp IDs are required and cannot be removed from the generated ID.

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

The following example shows that each record in the input is assigned a sequential record ID in the output.

Record	RecordID
John Smith	0
Mary Smith	1
Jane Doe	2
John Doe	3

The Unique ID stage produces a field named RecordID which contains the unique ID. You can rename the RecordID field as required.

Defining a Unique ID

By default, the Unique ID Generator stage creates a sequential ID, with the first record having an ID of 0, the second record having an ID of 1, the third record having an ID of 2, and so on. If you want to change how the unique ID is generated, follow this procedure.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Choose the method you want to use to generate the unique ID.

Options	Description
Sequential Numeric tag starting at	Assigns an incremental numeric value to each record starting with the number you specify. If you specify 0, the first record will have an ID of 0, the second record will have an ID of 1, and so on.

Options	Description
Sequential Numeric tag starting at value in a database field	

Options

Description

Assigns an incremental numerical value to each record starting with the maximum number read from the database field. This number is then incremented by 1 and assigned to the first record. For example, if the number read from the database field is 30, the first record will have an ID of 31, the second record will have an ID of 32, and so on.

Connection Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database, or modify or delete an existing connection, click **Manage**.

If you are adding or modifying a database connection, complete these fields:

Connection name

Enter a name for the connection. This can be anything you choose.

Database driver

Select the appropriate database type.

Connection options

Specify the host, port, instance, user name, and password to use to connect to the database.

Table view Specifies the table or view in the database that you want to query.

Database field Select a column from the list to generate a unique key.

The supported datatypes for unique ID generation are:

long A numeric data type that contains both negative and positive whole numbers between -2^{63} ($-9,223,372,036,854,775,808$) and $2^{63}-1$ ($9,223,372,036,854,775,807$).

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} ($-2,147,483,648$) and $2^{31}-1$ ($2,147,483,647$).

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52})\times 2^{1023}$. In E notation, the range of values is $-1.79769313486232E+308$ to $1.79769313486232E+308$.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23})\times 2^{127}$. In E notation, the range

Options	Description
	of values -3.402823E+38 to 3.402823E+38.
Date/Time stamp	Creates a unique key based on the date and time stamp instead of sequential numbering.
UUID	Creates a universally unique 32-digit identifier key for each record. The digits in the key are displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). Example: 123e4567-e89b-12d3-a456-432255330000
Off	Select this option only if you want to generate a non-unique key using an algorithm.

3. Click **OK**.

Using Algorithms to Augment a Unique ID

Unique ID Generator generates a unique ID for each record by either numbering each record sequentially or generating a date/time stamp for each record. You can optionally use algorithms to append additional information to the sequential or date/time unique ID, thereby creating a more complex unique ID and one that is more likely to be truly unique.

1. In the Unique ID Generator stage, click **Add**.
2. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID. One of the following:

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.

Metaphone (Spanish)	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
Metaphone 3	Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.
Nysiis	Phonetic code algorithm that matches an approximate pronunciation to an exact spelling and indexes words that are pronounced similarly. Part of the New York State Identification and Intelligence System. Say, for example, that you are looking for someone's information in a database of people. You believe that the person's name sounds like "John Smith", but it is in fact spelled "Jon Smyth". If you conducted a search looking for an exact match for "John Smith" no results would be returned. However, if you index the database using the NYSIIS algorithm and search using the NYSIIS algorithm again, the correct match will be returned because both "John Smith" and "Jon Smyth" are indexed as "JAN SNATH" by the algorithm.
Phonix	Preprocesses name strings by applying more than 100 transformation rules to single characters or to sequences of several characters. 19 of those rules are applied only if the character(s) are at the beginning of the string, while 12 of the rules are applied only if they are at the middle of the string, and 28 of the rules are applied only if they are at the end of the string. The transformed name string is encoded into a code that is comprised by a starting letter followed by three digits (removing zeros and duplicate numbers). This option was developed to respond to limitations of Soundex; it is more complex and therefore slower than Soundex.
Soundex	Returns a Soundex code of selected fields. Soundex produces a fixed-length code based on the English pronunciation of a word.
Substring	Returns a specified portion of the selected field.

3. In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
4. If you selected the substring algorithm, specify the portion of the field you want to use in the substring:
 - a) In the **Start position** field, specify the position in the field where you want the substring to begin.
 - b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have the following data in a field named LastName:

Augustine

If you specified 3 as the start position and 6 as the end position, the substring would produce:

```
gustin
```

5. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
6. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
7. Click **OK** to save your settings.
8. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Defining a Non-Unique ID

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Select **Off**.

This turns off the unique ID portion of the ID generation rules. With this option off, only the algorithm you choose in the following steps will be used to create the ID. This means that any records that have the same data in the fields you use to generate the ID will have the same ID. You can then use the ID for matching.

3. Click **OK**.
4. At the warning prompt, click **Yes**.
5. In the Unique ID Generator stage, click **Add**.
6. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID. One of the following:

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that

they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.

MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
Metaphone (Spanish)	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
Metaphone 3	Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.
Nysiis	Phonetic code algorithm that matches an approximate pronunciation to an exact spelling and indexes words that are pronounced similarly. Part of the New York State Identification and Intelligence System. Say, for example, that you are looking for someone's information in a database of people. You believe that the person's name sounds like "John Smith", but it is in fact spelled "Jon Smyth". If you conducted a search looking for an exact match for "John Smith" no results would be returned. However, if you index the database using the NYSIIS algorithm and search using the NYSIIS algorithm again, the correct match will be returned because both "John Smith" and "Jon Smyth" are indexed as "JAN SNATH" by the algorithm.
Phonix	Preprocesses name strings by applying more than 100 transformation rules to single characters or to sequences of several characters. 19 of those rules are applied only if the character(s) are at the beginning of the string, while 12 of the rules are applied only if they are at the middle of the string, and 28 of the rules are applied only if they are at the end of the string. The transformed name string is encoded into a code that is comprised by a starting letter followed by three digits (removing zeros and duplicate numbers). This option was developed to respond to limitations of Soundex; it is more complex and therefore slower than Soundex.
Soundex	Returns a Soundex code of selected fields. Soundex produces a fixed-length code based on the English pronunciation of a word.
Substring	Returns a specified portion of the selected field.

- In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
- If you selected the substring algorithm, specify the portion of the field you want to use in the substring:

- a) In the **Start position** field, specify the position in the field where you want the substring to begin.
- b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have the following data in a field named LastName:

```
Augustine
```

If you specified 3 as the start position and 6 as the end position, the substring would produce:

```
gustin
```

9. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
10. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
11. Click **OK** to save your settings.
12. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Write to Cache

Write to Cache loads output from a dataflow into a global cache, making the data available for lookup from the Query Cache stage. Using a global cache for data lookups improves performance compared to lookups to databases.

A global cache is system-wide, shared cache that will reside in memory. Choose a global cache if you want the cache to be available to multiple dataflows or when data does not change often or remains relatively static and when storage is not limited. A global cache is static as you can write to it only once. The cache can not be updated once it has been created.

Note: Write to Cache overwrites the cache each time the dataflow runs.

General

Option Name	Description
Cache name	Specifies the name you want to give to the cache. If there are caches already on the system, they are listed and you can choose one if you want to populate the existing cache with new data. To create a new cache, type the name you want for the new cache. The name must begin with a letter. It can contain an underscore but no other special characters. The name can contain numeric values.
Cache Fields	This column lists the field names that will be used in the cache. If you wish to change a field name, click the field name and enter a new name.
Stage Fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Type	This column lists the data type of each field.
Include	Check the box in this column to have the field written to the cache. Clear the box if you do not want the field written to the cache.
Key Field	<p>Check the box in this column if you want the field to be used as a key in the Query Cache stage. For example, if you have a dataflow field named AccountNumber and you want the Query Cache stage to look up data by querying for a matching value in the AccountNumber field, you would check the box in the Key Field column for the AccountNumber field.</p> <p>The fields you specify as key fields are available for selection in the Query Cache stage as key fields.</p>

Clearing a Global Cache

To clear a global cache you must create and execute a process flow. The process flow must contain a Clear Cache activity. The Clear Cache activity clears the global cache but does not delete it. You can also clear the cache automatically by scheduling a process flow.

Note: Please see *Dataflow Designer Guide* for instructions on how to create and schedule a process flow.



To manually clear the global cache data, follow these steps:

1. Drag the **Clear Cache** activity to the canvas.
2. Drag the **Success** activity to the canvas.
3. Connect the two activities.
4. Double-click the **Clear Cache** activity.
5. Select the cache. You can also select multiple caches to clear their data.
The caches that you create in Write to Cache stage are listed in Clear Cache activity.
6. Run the process flow.

Write to DB

The Write to DB stage writes the output of a dataflow to a database.

Note: Significant performance improvements can be achieved by using multiple runtime instances of Write to DB. To specify multiple runtime instances, click the **Runtime** button.

General Tab

Option Name	Description
Connection	Select the connection for the database you want to use in the Connection field. To make a new database connection, click Manage . For more information on creating database connections, see Database Connection Manager .
Table/View	After selecting a connection, specify the table or view to write to. Click the browse button ([...]) to navigate to the table or view that you want to use, or click Create Table to create a new table in the database. Note: If you are writing to a SQL database, you cannot write to views that reference more than one table. This is due to a limitation in SQL Server.

Option Name	Description
Create Table	<p>Creates a new table in the selected database. Choose the owner for the table in the Table owner field and specify the name for the new table in the Table name field. Table names are case sensitive. Specify a primary key by selecting a checkbox in the Primary key column. Also, specify the fields you want to write to the new table by checking the box in the Include column. Width column specifies the field's length for string data type. By default it is 512. If the column Allow Null is checked and the Input Fields contain a null value, then the dataflow will write the null value in the database. Note that you can edit the column name by changing the value in the Output Fields column.</p> <p>The Create Table button supports table creation in the following databases:</p> <ul style="list-style-type: none"> • Axion • DB2 • Derby/Cloudscape • Firebird • HSQLDB • Interbase • MaxDB/SapDB • McKoi • MySQL • Oracle • PostgreSQL • SQL Server • Sybase <p>Note: For DB2 databases, if you try to create a table and the page size is smaller than the total length of all string columns, you will get an error that says "Failed to build body from content. Serializable class not available to broker."</p>
Stage Fields	<p>In the Stage Fields column you can specify the field you want to write to the database field shown in the Database Field column.</p>
Include	<p>The Include column allows you to select the fields you want to write to.</p> <p>Note: To prevent poor performance you should have a sorted index or key in the database table.</p>

Note: The **Write to DB** stage writes all values of the `date` datatype as `String` values. This is the behavior of the *jTDS driver*, which is the default driver used by Spectrum. To handle all `date` datatype values as is, use Microsoft's JDBC driver.

Database Connection Manager

The Database Connection Manager allows you to manage registered database connections. To add, modify, delete, and test connections:

1. In the **Write To DB Options** dialog box, click **Manage**.
2. Click **Add**, **Modify**, or **Delete**.
3. If you are adding or modifying a database connection, complete these fields:
 - Connection name—Enter the name of the new connection.
 - Database driver—Select the appropriate database type.
 - Connection Options—Specify all the options, typically host, port, instance, user name, and password.

Note: You can test the connection by clicking **Test**.

4. If you are deleting a database connection, select the connection you want to remove and click **Delete**.

Runtime Tab

Option Name	Description
Write Mode	<p>Specifies the type of actions to take when writing to the database. One of the following:</p> <p>Insert Insert new records into the database but do not update existing records. This is the default setting.</p> <p>Update Update existing records in the database but do not insert new records.</p> <p>Note: If you select Update, the primary key column name used in the input table must match the primary key column name in the output table. If you try to update a table where the primary key column name does not match the input, or where the primary key column is not defined, the update will not work.</p> <p>Insert if not able to update Insert new records into the database if the record does not exist, otherwise update the existing record.</p>
Batch commit	<p>Select this option to commit changes to the database after a specified number of records are processed. By default this option is not selected, which means that changes are committed after each record is processed. Selecting this option can significantly improve the performance of the Write to DB stage.</p>

Option Name	Description
Batch size	<p>If you enable the Batch commit option, specifies the number of records to commit to the database in each batch. The default is 1,000. For dataflows created in Spectrum™ Technology Platform 7.0 and earlier, the default is 100.</p> <p>A larger batch size does not always offer better load performance. Consider the following factors when choosing a batch size:</p> <ul style="list-style-type: none"> • Data arrival rate to Write To DB stage: If data is arriving at slower rate than the database can process then modifying batch size will not improve overall dataflow performance. For example, dataflows with address validation or geocoding may not benefit from an increased batch size. • Network traffic: For slow networks, increasing batch size to a medium batch size (1,000 to 10,000) will result in better performance. • Database load and/or processing speed: For databases with high processing power, increasing batch size will improve performance. • Multiple runtime instances: If you use multiple runtime instances of the Write to DB stage, a large batch size will consume a lot of memory, so use a small or medium batch size (100 to 10,000). • Database roll backs: Whenever a statement fails, the complete batch is rolled back. The larger the batch size, the longer it will take to perform the to rollback.
Commit at the end	<p>Select this option to ensure that the commit to database operation occurs after all the records are transferred to the database.</p>
Batch count to commit	<p>Specify a value after which the records are to be committed. Records are committed to the database after every (batch count to commit * batch size) number of records are transferred to the database. For example, if Batch size is set as 1000 and Batch count to commit is set as 3, then the commit occurs after every 3000 records are transferred to the database.</p>
Truncate table before inserting data	<p>Select this option if you want to clear all data from the table before writing to the database.</p>
Drop and recreate the table if it already exists	<p>Select this option to delete and recreate the table before writing the dataflow's output to the table. This option is useful if you want the table's schema to match the fields from the dataflow and not contain any extraneous schema information.</p> <p>The table that will be deleted and recreated is the one specified in the Table/View field on the General tab. For example, if you specify the Customers table in the Table/View field, and you select Drop and recreate the table if it already exists, then the Customers table will be deleted from the database, and a new table named Customers will be created with a schema that matches the actual fields written to the table.</p>

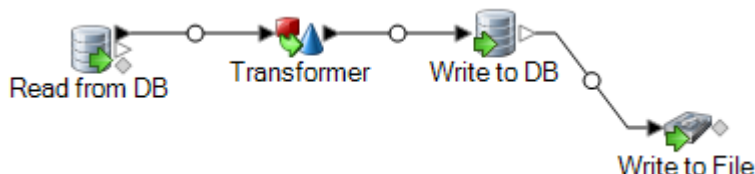
Configuring Error Handling in Write to DB

The Write to DB stage has an error port which allows you to filter out records that cause database errors when writing the record to a database, such as a primary key constraint violation or a unique constraint violation. These records can then be routed along another path in the dataflow while other records are successfully committed. For example, if you are processing 100 records and records 4, 23, and 56 cause a database error, these three records would be routed through the error port while the other 97 records would be committed to the database.

Note: Using the error port is optional. If you do not use the error port, the job will fail if any record causes an error.

- From the palette, choose the type stage you want to handle error records (for example, Write to File) and drag it onto the canvas. You have a couple options for selecting a stage:
 - To write failed records to a file, drag one of the following onto the canvas: Write to File, Write to XML, or Write to Variable Format File,.
 - To simply discard failed records, drag Write to Null onto the canvas.
- Connect the error port on Write to DB to the stage you want to handle failed records.

The following example shows the error port on Write to DB connected to a Write to File stage. In this example, records that cause an error when written to the database are instead written to the file specified in the Write to File stage.



When you run the dataflow, records that cause an error are routed through the error port. The records from the error port contain the fields specified in Write to DB plus the following fields:

Error.code	This field contains the numeric error code returned from the database. For example, given the error <code>ORA-00001: unique constraint ANKUSH.SYS_C0010018) violated</code> , the value in the Error.code field would be 1. See your database software's documentation for a listing of error codes.
Error.Message	This field contains the error message returned from the database. For example: <code>ORA-01034 ORACLE not available</code> . In this case, <code>ORACLE not available</code> would be the value in the Error.Message field. See your database software's documentation for a listing of error messages.
Error.SQLState	This field contains the SQLSTATE code which provides detailed information about the cause of the error. For a listing of SQLSTATE codes, see your database software's documentation.

Timestamp	The date and time on the Spectrum™ Technology Platform server when the error occurred.
Username	The name of the Spectrum™ Technology Platform user that ran the dataflow.

Write to File

Write to File writes dataflow output to a flat file. The records all contain the same fields. If you want to write records of varying format, see [Write to Variable Format File](#) on page 266. If you want to write records to an XML file, see [Write to XML](#) on page 276.

Tip: You can copy your source and paste it as the sink into your dataflow to quickly set up the file and use the same fields as you defined in your source.

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.</p> <p>While writing a file to an HDFS file server, the below compression formats are supported:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: Include the appropriate extension in the file name, to indicate the desired compression format to be used while writing the file.</p> <p>Attention: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>

Field Name	Description
Record type	<p>The format of the records in the file. Select one of:</p> <ul style="list-style-type: none"><li data-bbox="553 422 1425 514">Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.<li data-bbox="553 533 1425 625">Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.<li data-bbox="553 644 1425 766">Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.

Field Name	Description
Character encoding	The text file's encoding. Select one of these: <ul style="list-style-type: none"> UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. US-ASCII A character encoding based on the order of the English alphabet. UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first). UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first). ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1. ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3. ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5. CP850 An ASCII code page used to write Western European languages. CP500 An EBCDIC code page used to write Western European languages. Shift_JIS A character encoding for the Japanese language. MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions. CP1047 An EBCDIC code page with the full Latin-1 character set.

Field Name	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <p>Unix (U+000A) A line feed character separates the records. This is the standard record separator for Unix systems.</p> <p>Macintosh (U+000D) A carriage return character separates the records. This is the standard record separator for Macintosh systems.</p> <p>Windows (U+000D U+000A) A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</p> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>

Field Name	Description
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum™ Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Record length	<p>For fixed width files, specifies the exact number of characters in each record.</p> <p>For line sequential files, specifies the length, in characters, of the longest record in the file.</p>
First row is header record	<p>Specifies whether the first record in a delimited file contains header information and not data.</p> <p>For example, this file snippet shows a header row in the first record.</p> <pre>"AddressLine1" "City" "StateProvince" "PostalCode" "7200 13TH ST" "MIAMI" "FL" "33144" "One Global View" "Troy" "NY" 12180</pre>
Treat records with fewer fields than defined as malformed	<p>Delimited file records containing fewer fields than are defined on the Fields tab will be treated as malformed.</p>
Import	<p>Imports the file layout definition, encoding setting, and sort options from a settings file. The settings file is created by exporting settings from another Read from File or Write to File stage that used the same input file or a file that has the same layout as the file you are working with.</p>
Export	<p>Saves the file layout definition, encoding setting, and sort options to a settings file. You can then import these settings into other Read from File or Write to File stages that use the same input file or a file that has the same traits as the file you are working with now. You can also use the settings file with job executor to specify file settings at runtime.</p> <p>For information about the settings file, see The File Definition Settings File on page 147.</p>

Fields Tab

The Fields tab defines the names, positions, and, for fixed width and line sequential files, lengths of fields in the file. For more information, see the following topics:

[Defining Fields In a Delimited Output File](#) on page 238

[Defining Fields In a Line Sequential or Fixed Width File](#) on page 240

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the output records before they are written to the output file. Sorting is optional. For more information, see [Sorting Output Records](#) on page 243.

Runtime Tab

Option Name	Description
File name	This displays the file defined on the File Properties tab.
Generate multiple files	<p>Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum™ Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Management Console.</p> <p>Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.</p>
File path field	Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. This field is only enabled if you select Generate multiple files .
Write Mode	<p>Specifies whether to add the dataflow's output to the end of the file or to delete the existing data in the file before writing the output. One of the following:</p> <p>Overwrite Replaces the existing data in the output file each time the dataflow runs.</p> <p>Append Adds the dataflow's output to the end of the file without erasing the file's existing data.</p>

Defining Fields In a Delimited Output File

In the Write to File stage, the **Fields** tab defines the names, position, and, for some file types, lengths of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields.

If the output file contains a header record, you can quickly define the fields by clicking **Regenerate**.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps:

1. Click **Add**.
2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum™ Technology Platform supports the following data types:

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum™ Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).

string A sequence of characters.

time A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

4. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:
 - a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
 - b) In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

5. Click **Add**.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and the position of the remaining fields will be adjusted accordingly.
Modify	Modifies the field's name and type.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Defining Fields In a Line Sequential or Fixed Width File

In the Write to File stage, the **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

To add fields manually from a list of fields used in the dataflow, follow this procedure:

1. Click **Add**.
2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum™ Technology Platform supports the following data types:

- bigdecimal** A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.
- boolean** A logical type with two values: true and false.
- bytearray** An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum™ Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre> <Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names> </pre> <p>It is important to note that the Spectrum™ Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum™ Technology Platform list data type is similar to an XML complex data type.</p>
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

4. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Management Console, or it is the dataflow's default format specified in the type conversion options in Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:
 - a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, your selection

will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."

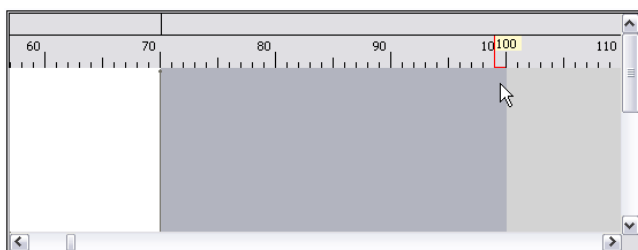
- b) In the **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in [Date and Time Patterns](#) on page 283. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 286.

- The **Start Position** and **Length** fields are automatically filled in based on the data in the dataflow and number of fields you have already added.
- Click **Add**.

Alternatively, you can also add a field by first defining the starting position and length of the field. To do this, under **Sample File** click at the position where you want to begin a field and drag to the left so that the desired field is highlighted, as shown here:



After defining the fields in your output file, you can edit its contents and layout. The **Recalculate start position** option tells the Write to File stage to recalculate the positions of the fields when you modify, move, or remove a field in the output file. Uncheck this box if you do not want the positions recalculated and instead want the fields to stay in their existing position after you edit the output file.

Option Name	Description
Add	Adds a field to the output.
Modify	Modifies the field's name, type, start position, and length.
Remove	Removes the selected field from the output.

Option Name	Description
Move Up/Move Down	Reorders the selected field.

Sorting Output Records

In the Write to File stage, the **Sort Fields** tab defines fields by which to sort the output records before they are written to the output file. Sorting is optional.

1. In Write to File, click the **Sort Fields** tab.
2. Click **Add**.
3. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields in the dataflow.
4. In the **Order** column, select Ascending or Descending.
5. Repeat until you have added all the output fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.
6. Default sort performance options for your system are set in the Management Console. If you want to override your system's default sort performance options, click **Advanced**. The **Advanced Options** dialog box contains the following sort performance options:

In memory record limit Specifies the maximum number of data rows a sorter will hold in memory before it starts paging to disk. By default, a sort of 10,000 records or less will be done in memory and a sort of more than 10,000 records will be performed as a disk sort. The maximum limit is 100,000 records. Typically an in-memory sort is much faster than a disk sort, so this value should be set high enough so that most of the sorts will be in-memory sorts and only large sets will be written to disk.

Note: Be careful in environments where there are jobs running concurrently because increasing the **In memory record limit** setting increases the likelihood of running out of memory.

Maximum number of temporary files Specifies the maximum number of temporary files that may be used by a sort process. Using a larger number of temporary files can result in better performance. However, the optimal number is highly dependent on the configuration of the server running Spectrum™ Technology Platform. You should experiment with different settings, observing the effect on performance of using more or fewer temporary files. To calculate the

approximate number of temporary files that may be needed, use this equation:

$$\frac{(NumberOfRecords \times 2)}{NumberOfTempFiles} = InMemoryRecordLimit$$

Note that the maximum number of temporary files cannot be more than 1,000.

Enable compression

Specifies that temporary files are compressed when they are written to disk.

Note: The optimal sort performance settings depends on your server's hardware configuration. Nevertheless, the following equation generally produces good sort performance:

$$(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$$

The File Definition Settings File

A file definition settings file contains the file layout, encoding, and sort options that have been exported from a Read from File or Write to File stage. The file definitions settings file can be imported into Read from File or Write to File to quickly set the stage's options instead of manually specifying the options.

The easiest way to create a file definition settings file is to use specify the file settings using Read from File or Write to File, then click the **Export** button to generate the file definitions settings file.

However, for your information the schema of the file definition settings file is shown below. Each element in the XML file has a type, and if that type is anything other than string or integer, the acceptable values are shown. These values correspond directly to options in the stage's dialog box. For example, the FileTypeEnum element corresponds to the Record Type field on the File Properties tab, and the following three values are shown in the schema: linesequential, fixedwidth, and delimited.

Note: If you enter "custom" for the LineSeparator, FieldSeparator or TextQualifier fields, a corresponding custom element must also be included (for example, "CustomLineSeparator", "CustomFieldSeparator", or "CustomTextQualifier") with a hexadecimal number representing the character, or sequence of characters, to use.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FileSchema" nillable="true" type="FileSchema"/>
  <xs:complexType name="FileSchema">
```

```

<xs:sequence>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="linesequential"
    name="Type"
    type="FileTypeEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="UTF-8" name="Encoding" type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="RecordLength"
    type="xs:int"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="default"
    name="LineSeparator"
    type="LineSeparatorEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomLineSeparator"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="comma"
    name="FieldSeparator"
    type="FieldSeparatorEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomFieldSeparator"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="none"
    name="TextQualifier"
    type="TextQualifierEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomTextQualifier"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="false"

```

```

        name="HasHeader"
        type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="true"
  name="EnforceColumnCount"
  type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="Fields"
  type="ArrayOfFieldSchema"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="FileTypeEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="linesequential"/>
    <xs:enumeration value="fixedwidth"/>
    <xs:enumeration value="delimited"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="LineSeparatorEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="default"/>
    <xs:enumeration value="windows"/>
    <xs:enumeration value="unix"/>
    <xs:enumeration value="mac"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FieldSeparatorEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="comma"/>
    <xs:enumeration value="tab"/>
    <xs:enumeration value="space"/>
    <xs:enumeration value="semicolon"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="pipe"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TextQualifierEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="single"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ArrayOfFieldSchema">
  <xs:sequence>
    <xs:element

```

```

        minOccurs="0"
        maxOccurs="unbounded"
        name="Field"
        nillable="true"
        type="FieldSchema"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="FieldSchema">
    <xs:sequence>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Name"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="string"
            name="Type"
            type="xs:string"/>
        <xs:element
            minOccurs="1"
            maxOccurs="1"
            name="Position"
            type="xs:int"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Length"
            type="xs:int"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="false"
            name="Trim"
            type="xs:boolean"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Locale"
            type="Locale"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Pattern"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="none"
            name="Order"
            type="SortOrderEnum"/>
    </xs:sequence>

```

```

</xs:complexType>
<xs:complexType name="Locale">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Country"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Language"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Variant"
      type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="SortOrderEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="ascending"/>
    <xs:enumeration value="descending"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Configuring Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for Write to File stage.

1. Open the dataflow in Enterprise Designer.
2. If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
3. Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
4. Click **Add**. The **Define Dataflow Options** dialog box appears.
5. Expand the **Write to File** stage.

The Dataflow options exposed are:

1. Character Encoding
2. Field Separator
3. Text Qualifier

4. Record Length
5. First Row is Header Record
6. The selected Write to File option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.
7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Dataflow Options Rules

1. *Character Encoding*: All encoding types that are valid for the underlying JVM are accepted. This option cannot be empty.
2. *Field Separator*: Any single character delimiter is accepted. Currently, HEX values and spaces are not supported.
3. *Text Qualifier*: Any single character qualifier is accepted. HEX values are not supported.
4. *Record Length*: Only integers accepted. Cannot be blank or non numeric.
5. *Starting Record*: Only integers accepted. Cannot be non numeric.
6. *Max records*: Only integers accepted. Cannot be non numeric.
7. *First Row is Header*: Only boolean values of `true` and `false` accepted. Cannot be blank.

Write to Hadoop Sequence File

The Write to Hadoop Sequence File stage writes data to a sequence file as output from a dataflow. A sequence file is a flat file consisting of binary key/value pairs. For more information, go to wiki.apache.org/hadoop/SequenceFile.

Note: The Write to Hadoop Sequence File stage only supports delimited, uncompressed sequence files located on Hadoop Distributed File System (HDFS).

File Properties Tab

Fields	Description
Server	Indicates the file you select in the File name field is located on the Hadoop system. You need to create a connection to the Hadoop file server in the Management Console before using it in the stage. If you select a file on the Hadoop system, the server name will be the name you specify in the Management Console while creating a file server.
File name	Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier:</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>

Fields Tab

The Fields tab defines the names, positions, and types of fields in the file. For more information, see [Defining Fields In an Output Sequence File](#) on page 251

Defining Fields In an Output Sequence File

In the Write to Hadoop Sequence File stage, the **Fields** tab defines the names, positions, and types of fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

1. To select the desired fields from the input data, or an existing file, click **Quick Add**.

- a) Select the specific fields from the input data.
- b) Click **OK**.

2. To add new fields, click **Add**.

- a) Enter the **Name** of the field.
- b) Select the **Type** of the field. The stage supports the following data types:

boolean	A logical type with two values: true and false.
date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

Note: In Parquet files, `datetime` and `time` datatypes are mapped as `String`. In RC files, `datetime` datatype is mapped as `timestamp`.

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
bigdecimal	A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

Note: For RC, Avro, and Parquet Hive files, the `bigdecimal` datatype is converted to a `decimal` datatype with precision 38 and scale 10.;

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).

Note: In RC files, the `long` datatype is mapped as `bigint` datatype.

string A sequence of characters.

- c) In the **Position** field, enter the position of this field within the record.

For example, in this input file, `AddressLine1` is in position 1, `City` is in position 2, `StateProvince` is in position 3, and `PostalCode` is in position 4.

```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|"12180"
```

- If you're overwriting an existing file, click **Regenerate** to pick the schema from the existing file, and then modify it.
This generates the schema based on the first 50 records in the input data to this stage.
- If you want to have any excess space characters removed from the beginning and end of a field's character string, select the **Trim Spaces** check box.
- Specify one of the following options to generate the key:

Auto Generate The Hadoop cluster auto generates the key. For auto generation, all the fields in the grid are considered value fields. The data type of the key is `long`.

User Defined By default, the first field in the grid is selected as the key field. An icon is displayed to indicate that the field is the key field. You can select any other field as the key field.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and the position of the remaining fields will be adjusted accordingly.
Modify	Modifies the field's name and type.

Option Name	Description
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Write to Hive File

The **Write to Hive File** stage writes the dataflow input to the specified output Hive file.

You can select any of the below supported Hive file formats for the output file:

- ORC
- RC
- Parquet
- Avro

File Properties tab

Table 6: Common File Properties

Fields	Description
Server name	Indicates that the file selected in the File name field is located on the Hadoop system. Once you select a file located on a Hadoop system, the Server name reflects the name of the respective file server, as specified in Management Console.
File name	Click the ellipses button (...) to browse to the output Hive file to be created in the defined Hadoop file server. The output data of this stage is written to the selected file. Note: You need to create a connection to the Hadoop file server in Management Console before using it in the stage.
File type	Select one of the four supported Hive file formats: <ul style="list-style-type: none"> • ORC • RC • Parquet • Avro

Table 7: ORC File Properties

Fields	Description
Buffer size	<p>Defines the buffer size to be allocated while writing to an ORC file. This is specified in kilobytes.</p> <p>Note: The default buffer size is 256 KB.</p>
Stripe size	<p>Defines the size of stripes to be created while writing to an ORC file. This is specified in megabytes.</p> <p>Note: The default stripe size is 64 MB.</p>
Row index stride	<p>Defines the number of rows to be written between two consecutive row index entries.</p> <p>Note: The default Row Index Stride is 10000 rows.</p>
Compression type	<p>Defines the compression type to be used while writing to an ORC file. The compression types available are <i>ZLIB</i> and <i>SNAPPY</i>.</p> <p>Note: The default compression type is <i>ZLIB</i>.</p>
Padding	<p>Indicates whether the stripes are padded to minimize stripes that cross HDFS block boundaries, while writing to an ORC file.</p> <p>Note: By default, the Padding checkbox is selected.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in the Preview grid, after the dataflow is run at least once and the data has been written to the selected file.</p>

Table 8: RC File Properties

Fields	Description
Buffer size	<p>Defines the buffer size to be allocated while writing to an RC file. This is specified in kilobytes.</p> <p>Note: The default buffer size is 256 KB.</p>

Fields	Description
Block size	<p>Defines the size of blocks to be created while writing to an RC file. This is specified in megabytes.</p> <p>Note: The default block size is 64 MB.</p>
Compression type	<p>Defines the compression type to be used while writing to an RC file. The compression types available are NONE and DEFLATE.</p> <p>Note: The default compression type is NONE.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in the Preview grid, after the dataflow is run at least once and the data has been written to the selected file.</p> <p>The Fields tab is used to define the sequence and datatype of the required fields.</p> <p>Note: For RC file type, you must define the metadata of the output file before clicking Preview to load the Preview grid.</p>

Table 9: Parquet File Properties

Fields	Description
Compression type	<p>Defines the compression type to be used while writing to a PARQUET file. The compression types available are UNCOMPRESSED, GZIP and SNAPPY.</p> <p>Note: The default compression type is UNCOMPRESSED.</p>
Block size	<p>Defines the size of block to be created while writing to a PARQUET file. This is specified in megabytes.</p> <p>Note: The default block size is 128 MB.</p>
Page size	<p>The page size is for compression. When reading, each page can be decompressed independently. This is specified in kilobytes.</p> <p>Note: The default page size is 1024 KB.</p>

Fields	Description
Enable dictionary	To enable/disable dictionary encoding. Attention: The dictionary must be enabled for the Dictionary Page Size to be enabled. Note: The default value is <code>true</code> .
Dictionary Page size	There is one dictionary page per column per row group when dictionary encoding is used. The dictionary page size functions like the page size. This is specified in kilobytes. Note: The default dictionary Page size is 1024 KB.
Writer version	Parquet supports two writer API versions: <code>PARQUET_1_0</code> and <code>PARQUET_2_0</code> . Note: The default is <code>PARQUET_1_0</code> .
Preview	The first 50 records of the written file are fetched and displayed in the Preview grid, after the dataflow is run at least once and the data has been written to the selected file.

Table 10: Avro File Properties

Fields	Description
Sync Interval (in Bytes)	Specifies the approximate number of uncompressed bytes to be written in each block. The valid values range from 32 to 2^{30} . However, it is suggested to keep the sync interval between 2K and 2M. Note: The default sync interval is 16000.
Compression	Defines the compression type to be used while writing to an Avro file. The compression types available are NONE , SNAPPY and DEFLATE . Choosing DEFLATE compression gives you an additional option of selecting the compression level (described below). Note: The default compression type is NONE .

Fields	Description
Compression level	<p>This field is displayed if you select the <code>DEFLATE</code> option in the Compression field above.</p> <p>It can have values ranging from 0 to 9, where 0 denotes no compression. The compression level increases from 1 to 9, with a simultaneous increase in the time taken to compress the data.</p> <p>Note: The default compression level is 1.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in this grid, after the dataflow is run at least once and the data is written to the selected file.</p>

Fields tab

The **Fields** tab defines the names and types of the fields as present in the source file of this stage, and to be selected to be written to the output file.

For more information, see [Defining Fields for Writing to Hive File](#) on page 257.

Runtime tab

The **Runtime** tab provides the option to **Overwrite** an existing file of the same name in the configured Hadoop file server. If you check the **Overwrite** checkbox, then on running the dataflow, the new output Hive file overwrites any existing file of the same name in the same Hadoop file server.

By default, the **Overwrite** checkbox is unchecked.

Note: If you do not select **Overwrite**, an exception is thrown while running the dataflow, if the file to be written has the same name as an existing file in the same Hadoop file server.

Defining Fields for Writing to Hive File

In the **Fields** tab of the **Write to Hive File** stage, the schema names and datatypes of the fields in the input data to the stage are listed.

- To select the desired fields from the input data, or an existing file, click **Quick Add**.
 - Select the specific fields from the input data.
 - Click **OK**.
- To add new fields, click **Add**.
 - Enter the **Name** of the field.
 - Select the **Type** of the field. The stage supports the following data types:
 - boolean** A logical type with two values: true and false.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

Note: In Parquet files, `datetime` and `time` datatypes are mapped as `String`. In RC files, `datetime` datatype is mapped as `timestamp`.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.

integer A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

Note: For RC, Avro, and Parquet Hive files, the `bigdecimal` datatype is converted to a `decimal` datatype with precision 38 and scale 10.;

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).

Note: In RC files, the `long` datatype is mapped as `bigint` datatype.

string A sequence of characters.

- c) In the **Position** field, enter the position of this field within the record.

For example, in this input file, AddressLine1 is in position 1, City is in position 2, StateProvince is in position 3, and PostalCode is in position 4.

```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|"12180"
```

3. If you're overwriting an existing file, click **Regenerate** to pick the schema from the existing file, and then modify it.

This generates the schema based on the metadata of the existing file, in case of ORC and Parquet output files. For RC output files, you must add the fields explicitly to overwrite existing fields.

The **Name** column lists the names of the various columns of the input data. The **Type** column lists the datatypes of each respective field of the input data.

Note: In case of *Parquet* file type, another column **Nullable** indicates whether the field is nullable or not. You can check this checkbox for a particular field to make the field nullable, or uncheck it otherwise.

4. You can modify the names, datatypes and sequence of the selected columns in the output using the below buttons:

Option Name	Description
Add	Adds a field to the output.
Modify	Modifies the selected field's name and datatype.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the position of the selected field in the output.

5. Click **OK**.

Write to NoSQL DB

The Write to NoSQL DB writes the output of a dataflow to a NoSQL database. The stage supports the MongoDB and Couchbase database types.

Note: Significant performance improvements can be achieved by using multiple runtime instances of Write to NoSQLDB. To specify multiple runtime instances, click the **Runtime** button.

General Tab

Fields	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the NoSQL Connection Management in the Tools menu of Enterprise Designer. If you want to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>NoSQL database Select the appropriate database type.</p> <p>Username Enter the username to connect to the database.</p> <p>Note: For Couchbase, username is not mandatory. You can enter any username. The connection will be successful as long as you use the correct password that you supplied when creating the bucket.</p> <p>Password Enter the password to connect to the database.</p> <p>Hostname Specify the hostname on which the database runs.</p> <p>Port Specify the port to use to connect to the database.</p> <p>Database Specify the database from which the data is to be retrieved.</p> <p>Note: While the term Database is used in the user interface, Couchbase calls it a bucket.</p>
Table/View	<p>Specify the name of the collection you want to write to.</p> <p>You can create a new collection in the NoSQL database by entering a collection name in the Table/View drop box and clicking Create Table.</p> <p>Note: For Couchbase the 'table/view drop down' and 'create table' button is disabled, as we write to a bucket instead of a view. In addition, the Preview button is also disabled.</p>
Ignore NULL Values	<p>If this option is enabled, any field that has a NULL value, is ignored.</p> <p>Note: If you do not enable this option, any field that has a NULL value will be also written to the database.</p>

Fields	Description
Preview	<p>Displays the records from the selected table.</p> <p>Note: For MongoDB data sources, clicking Preview shows the filtered records, if one or more <code>where</code> clauses have been entered in the Where field. If no where clause has been entered, the preview displays all the records.</p> <p>Note: For Couchbase data sources, clicking Preview also displays the added <code>_id</code> field containing the key. If the record already has an <code>_id</code> field, then the added <code>_id</code> field overwrites the pre-existing one at the time of previewing the fields.</p>
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to write to the database. For more information, see [Defining Fields in a NoSQL Database](#) on page 261.

Defining Fields in a NoSQL Database

In the Write to NoSQL DB stage, the **Fields** tab defines the names and types of fields fetched from the previous stage.

1. On the **Fields Tab**, click **Regenerate**.

This displays the fields that are flowing from the previous stage.

The data is displayed in the following format: `Fieldname (datatype)`.

Note: For Couchbase, if the record has an `_id` field, this field will be used as a key for writing the record into the database. Else, the key for the record, will be autogenerated and written to the database.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.

- From the **Fields** tab, you can either individually select each field to be written to the database or click **Select All** to select all the fields.
- Optionally, from the **Runtime** tab, you can set the batch size. The batch size denotes the number of records that are to be written to the database at one time.
- Click **OK**.

NoSQL DB Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for NoSQL DB.

- Open the dataflow in Enterprise Designer.
- If you want to configure runtime options for a stage in an embedded dataflow, open the embedded dataflow.
- Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
- Click **Add**. The **Define Dataflow Options** dialog box appears.
- Expand the NoSQLDB stage.
- The Dataflow options are exposed as described in the following table:

Database	Read	Write
Mongo DB	Connection	Connection
	Table	Table
Couchbase DB	Connection	Connection
	View	
	Design Document Name	

The selected NoSQL DB option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
10. If you want to change the default value, specify a different value in the **Default value** field.

Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Enterprise Designer. Instead, you must use Management Console. For more information, see [Specifying Default Service Options](#).

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Write to Spreadsheet

Write to Spreadsheet writes data to an Excel spreadsheet as output from a dataflow.

File Properties Tab

The **File Properties** tab contains options for specifying the spreadsheet and data to write from a dataflow.

Field Name	Description
Server name	Indicates the file you select in the File name field is located on the Spectrum™ Technology Platformserver.

Field Name	Description
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.</p> <p>While writing a file to an HDFS file server, the below compression formats are supported:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: Include the appropriate extension in the file name, to indicate the desired compression format to be used while writing the file.</p> <p>Attention: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Write Mode	<p>Specifies how you want to write data in the spreadsheet in order to write into the dataflow. You can create a spreadsheet at runtime using any of the following options. Options are as following:</p> <p>Create or Overwrite Creates a new file and replaces the existing data in the output file each time the dataflow runs.</p> <p>Insert Adds the dataflow's output to the mapped area and shifts the data down if already present there.</p> <p>Append Adds the dataflow's output to the end of the file without erasing the file's existing data.</p>
Sheet name	Specifies a sheet name in the spreadsheet to which you want to write data into the dataflow.
Start writing from	Specifies either a row-column combination (A1 or B2 ..) or a column from where you want the data to be written. For the option Insert you need to provide both row & column. For the option Append you can only provide column as it ignores the row value.
First row as column header	Specifies the first row in a file contains header information and not data.

Fields Tab

The Fields tab defines columns, positions, types and nullable values for the fields in the file. For more information, see the following topic:

[Defining fields in an Output file](#) on page 265

Defining fields in an Output file

In Write to Spreadsheet, the **Fields** tab defines the names, position, and data types of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields. If the option **Nullable** is checked and the **Name** field contains a null value, then the dataflow will write the null value in the spreadsheet.

If the output file contains a header record, you can quickly define the fields by clicking **Regenerate**.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps:

1. Click **Add**.
2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum™ Technology Platform supports the following data types:

bigdecimal A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.

boolean A logical type with two values: true and false.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

date A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Management Console.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -3.402823E+38 to 3.402823E+38.

integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

4. Click **Add**.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and Write to Spreadsheet will adjust the remaining fields accordingly.
Modify	Modifies the field's name and type.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Write to Variable Format File

Write to Variable Format File writes records of varying layout to a file.

Variable format files have these characteristics:

- Records in the file may have different fields, and different numbers of fields.
- Each record must contain a tag (usually a number) identifying the type of record.
- Hierarchical relationships are supported.

Example of a Variable Format File

This example shows a variable format file containing information about checking account activity for two customers, Joe Smith and Anne Johnson. In this example, the file is a delimited file that uses a comma as the field delimiter.

```
001 Joe,Smith,M,100 Main St,555-234-1290
100 CHK12904567,12/2/2007,6/1/2012,CHK
200 1000567,1/5/2012,Fashion Shoes,323.12
001 Anne,Johnson,F,1202 Lake St,555-222-4932
100 CHK238193875,1/21/2001,4/12/2012,CHK
200 1000232,3/5/2012,Blue Goose Grocery,132.11
200 1000232,3/8/2012,Trailway Bikes,540.00
```

The first field in each record contains the tag which identifies the type of record and therefore the record's format:

- 001: Customer record
- 100: Account record
- 200: Account transaction record

For delimited files it is common for the tag value (001, 100, 200) to be in a fixed number of bytes at the start of the record as shown in the above example.

Each record has its own format:

- 001: FirstName,LastName,Gender,Address,PhoneNumber
- 100: AccountID,DateOpened,ExpirationDate,TypeOfAccount
- 200: TransactionID,DateOfTransaction,Vendor,Amount

Record format 100 (account record) is a child of the previous 001 record, and record format 200 (account transaction record) is a child of the previous record 100 (account record). So in the example file, Joe Smith's account CHK12904567 had a transaction on 1/5/2012 in the amount of 323.12 at Fashion Shoes. Likewise, Anne Johnson's account CHK238193875 had two transactions, one on 3/5/2012 at Blue Goose Grocery and one on 3/8/2012 at Trailway Bikes.

File Properties Tab

Option Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Enterprise Designer or on the Spectrum™ Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum™ Technology Platform.

Option Name	Description
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want.</p> <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Root tag name	<p>The tag to use for records that are a parent of other record types. For example if you have three record types 001, 100, and 200, and record types 100 and 200 are children of record type 001, then 001 is the root tag.</p>
Use fixed-width tags	<p>Specifies whether to allocate a fixed amount of space at the beginning of each record in which to place the record tag. For example, the following shows a file with the tags 001, 100, and 200 in a fixed-width field:</p> <pre>001 Joe, Smith, M, 100 Main St, 555-234-1290 100 CHK12904567, 12/2/2007, 6/1/2012, CHK 200 1000567, 1/5/2012, Mike's Shoes, 323.12</pre>
Tag width	<p>If you check the Use fixed-width tags box, this option specifies the number of spaces to allocate for tags at the beginning of each record. For example, if you specify 7, then the first seven positions in each record will be reserved for the tag. The value you specify must be greater than or equal to the size in characters of the longest tag name. For information on tag names, see Tag Names in Variable Format Files on page 275.</p> <p>The value in the Tag width field is automatically increased if you add fields on the Fields tab that have a name that is longer than the value specified.</p> <p>The maximum tag width is 1024.</p>
Remove numeric tag prefix	<p>Removes the "NumericTag_" portion of the field name before writing the tag to the file. The "NumericTag_" prefix is added to tag names by the Read from Variable Format File stage for any tag names that start with a number. This is because the tag name is used as the name of a list dataflow field which contains the data in the record, and dataflow field names cannot begin with a number. For example, a tag 100 would be changed to list field named "NumericTag_100". If you enable this option, this field would be written to the output file as a record with a tag of "100" instead of "NumericTag_100".</p>

Option Name	Description
Character encoding	The text file's encoding. Select one of these:
UTF-8	Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
UTF-16	Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
US-ASCII	A character encoding based on the order of the English alphabet.
UTF-16BE	UTF-16 encoding with big endian byte serialization (most significant byte first).
UTF-16LE	UTF-16 encoding with little endian byte serialization (least significant byte first).
ISO-8859-1	An ASCII character encoding typically used for Western European languages. Also known as Latin-1.
ISO-8859-3	An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.
ISO-8859-9	An ASCII character encoding typically used for Turkish language. Also known as Latin-5.
CP850	An ASCII code page used to write Western European languages.
CP500	An EBCDIC code page used to write Western European languages.
Shift_JIS	A character encoding for the Japanese language.
MS932	A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.
CP1047	An EBCDIC code page with the full Latin-1 character set.

Option Name	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file.</p> <p>For example, this record uses a pipe () as a field separator:</p> <pre data-bbox="553 430 1421 493">7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul data-bbox="553 556 690 751" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Tag separator	<p>Specifies the character placed after the tag field to demarcate the identifying field for each record in a delimited file. A tag separator must be a single character.</p> <p>By default, these characters are available to be selected as tag separators:</p> <ul data-bbox="553 1018 690 1213" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a tag separator, click the ellipses button to add and select a custom tag separator.</p> <p>Note: By default, the Record separator character is the same as the selected Field separator character. To enable this field and select a different character, uncheck the Same as Field separator checkbox.</p>
Same as Field separator	<p>Indicates if the tag separator is the same as the field separator. Uncheck this to select a different character as the tag separator.</p> <p>Note: By default, this checkbox is checked and the Tag separator field is disabled.</p>

Option Name	Description
Text qualifier	<p>The character used to surround text values in a delimited file. For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <p>Unix (U+000A) A line feed character separates the records. This is the standard record separator for Unix systems.</p> <p>Macintosh (U+000D) A carriage return character separates the records. This is the standard record separator for Macintosh systems.</p> <p>Windows (U+000D U+000A) A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</p> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum™ Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>

Fields Tab

The **Fields** tab controls which fields from the dataflow are included in the output file.

Option Name	Description
Add	<p>Click to add a field to the output.</p> <p>For information about constructing dataflow fields for use with Write to Variable Format File, see Writing Flat Data to a Variable Format File on page 273.</p>
Modify	<p>Click to modify the name of the tag. This button is only enabled when a tag is selected. If the Use fixed-width tags option is enabled on the File Properties tab, the tag width is automatically adjusted if you enter a longer tag name.</p> <p>Note: Using this button to modify the root tag name has the same effect as modifying the value of the Root tag name field on the File Properties tab.</p>
Remove	<p>Removes the selected field from the output. If you remove a list field all child fields are also removed. If you remove a child field, just the selected child field is removed from the list field.</p>
xx Remove All	<p>Removes all the fields from the output.</p>
Move Up/Move Down	<p>Reorders the selected field.</p>

Runtime Tab

Option Name	Description
File name	<p>This displays the file defined on the File Properties tab.</p>

Option Name	Description				
Generate multiple files	<p>Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum™ Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Management Console.</p> <p>Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.</p>				
File path field	<p>Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. Note that only the simple type elements mapped directly to a root tag will be listed in the File path field. This field is only enabled if you select the Generate multiple files.</p>				
Write Mode	<p>Specifies whether to add the dataflow's output to the end of the file or to delete the existing data in the file before writing the output. One of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">Overwrite</td> <td>Replaces the existing data in the output file each time the dataflow runs.</td> </tr> <tr> <td>Append</td> <td>Adds the dataflow's output to the end of the file without erasing the file's existing data.</td> </tr> </table>	Overwrite	Replaces the existing data in the output file each time the dataflow runs.	Append	Adds the dataflow's output to the end of the file without erasing the file's existing data.
Overwrite	Replaces the existing data in the output file each time the dataflow runs.				
Append	Adds the dataflow's output to the end of the file without erasing the file's existing data.				

Writing Flat Data to a Variable Format File

In a Spectrum™ Technology Platform dataflow each record has the same fields. However, in a variable format file, not all records contain the same fields. In order to write flat data from a dataflow to a variable format file you need to break up each record in the dataflow, grouping the fields from each record into list fields corresponding to the record types you want to use for the variable format file. A list field is a collection of fields. For example, the fields FirstName, LastName, Gender, Address, and Phone could be grouped together into a list field called AccountOwner.

To write flat data to a variable format file, use an Aggregator stage to group fields into list fields corresponding to the record types you want to write to the variable format file. To do this:

1. Place an Aggregator stage in your dataflow anywhere upstream from the Write to Variable Format File stage.
2. Double-click the Aggregator stage to open its options window.

3. Select **Group by** then click **Add**.
4. In the **Group By** field, select the field that contains a unique identifier that can be used to identify related data. This field's value should be unique across the records in the flat data. For example, an account number, a social security number, or a phone number.

Note: The field you select should be sorted. If it is not, use a Sorter stage to sort the records by the field.

5. Click **OK**.
6. Select **Output lists** then click **Add**.
Each output list will represent one record type in the variable format file.
7. Select **New data type** and in the **Type name** field specify the type of information that will be contained in this data type. This will become a record type in the variable format file. For example, this data type will contain records related to account transactions, you could name the type "AccountTransaction".
8. In the **Name** field, enter the name you want to give to this field. This may be the same name you specify in the **Type name** field.
9. Click **OK**.
10. Select the data type you just created and click **Add**.
11. Leave the option **Existing field** selected and select one of the fields you want to include in this data type then click **OK**. Remember that this will become a record type in the variable format file. Repeat to add additional fields to this record type.
12. Create additional output lists for each record type you want to have in the variable format file. When finished, click **OK** to close the Aggregator options.

The fields coming out of the Aggregator stage are now grouped into list fields that correspond to the record types you want to include in the variable format file output.

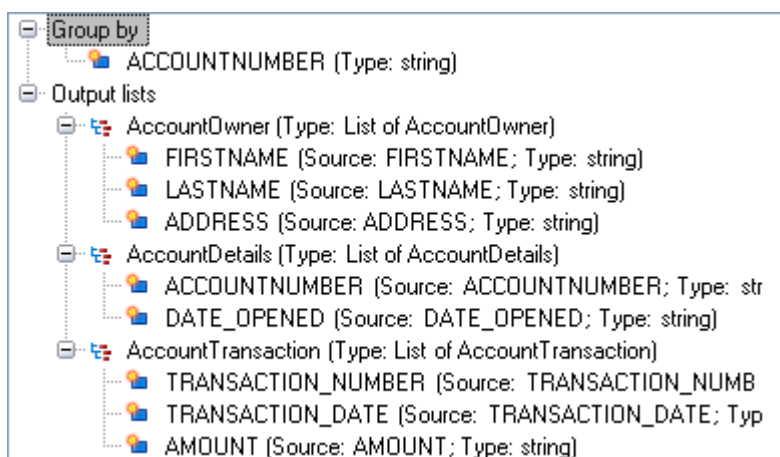
For example, given this flat data:

```
FIRSTNAME, LASTNAME, ADDRESS, ACCOUNTNUMBER, DATE_OPENED, TRANSACTION NUMBER, TRANSACTION DATE, AMOUNT
Joe, Smith, 100 Main St, CHK12904567, 12/2/2007, 1000567, 1/5/2012, 323.12
```

You would want to convert it to something like this in the variable format file:

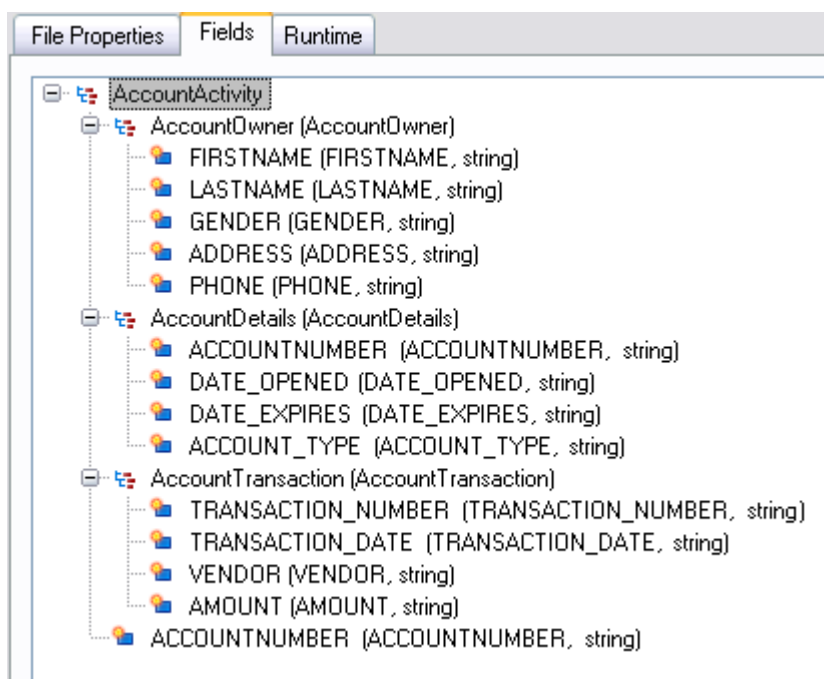
```
AccountOwner      Joe, Smith, 100 Main St
AccountInformation CHK12904567, 12/2/2007
Transaction       1000567, 1/5/2012, 323.12
```

To accomplish this, you would create an Aggregator stage that is configured like this:



Tag Names in Variable Format Files

In a variable format file, each record in the output file has a tag which indicates the record type. In Write To Variable Format File, the field name is used as the tag name in the output file. For example, consider these fields:



These fields would be written to the file as follows. Note that in this example the account has two AccountTransaction records.

```
AccountOwner      Anne,Johnson,F,1202 Lake St,555-222-4932
AccountDetails    CHK238193875,1/21/2001,4/12/2012,CHK
```

```
AccountTransaction 1000232,3/5/2012,Blue Goose Grocery,132.11
AccountTransaction 1000232,3/8/2012,Trailway Bikes,540.00
```

Note: Only list fields containing simple fields such as strings are written to the output file. If a list field consists only of other list fields it is not written to the output file. In the above example, no record with an AccountActivity tag would be written to the output file because AccountActivity consists only of other list fields (AccountOwner, AccountDetails, and AccountTransaction).

Write to XML

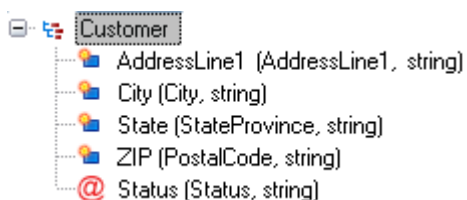
The Write to XML stage writes the output of a job or subflow to an XML file.

File Properties Tab

Field Name	Description
Data file	<p>Specifies the path to the output XML file. Click the ellipses button (...) to browse to the file you want.</p> <p>Note: If the Spectrum™ Technology Platform server is running on Unix or Linux, remember that file names and paths on these platforms are case sensitive.</p>
Actual File	<p>Displays the structure specified in the Fields tab. If you click an element and the file specified in the Data file field contains the element, a preview of the data will be displayed. Note that only data from simple elements can be displayed in the preview.</p>
Export Schema	<p>Click this button to save an XSD file that represents the schema shown in the Actual File view. The schema file is immediately saved to the location you specify.</p>

Fields Tab

The **Fields** tab defines the fields you want to include in the output XML file. When you add fields, they are displayed in a tree structure. The tree displays the name of the element or attribute that will be written to the XML file. In parentheses following the element/attribute name is the name of the dataflow field followed by the data type. For example:



This indicates that four elements and one attribute will be written to the XML file. The attribute is indicated by the red "@" sign.

Note that the element State will contain the data from the field StateProvince and be string data. Likewise, the element ZIP will contain data from the PostalCode field and be string data. The XML file might look like this:

```
<XmlRoot>
  <Customer Status="0">
    <AddressLine1>7713 Mullen Dr</AddressLine1>
    <City>Austin</City>
    <State>TX</State>
    <ZIP>78757-1346</ZIP>
  </Customer>
  <Customer Status="0">
    <AddressLine1>1825B Kramer Ln</AddressLine1>
    <City>Austin</City>
    <State>TX</State>
    <ZIP>78758-4260</ZIP>
  </Customer>
</XmlRoot>
```

Note: The root element name (in this example `<XmlRoot>`) is specified on the **File Properties** tab.

The following table describes the options on the **Fields** tab.

Option Name	Description
Add	Adds a field to the output.

Option Name	Description
Modify	<p data-bbox="553 373 1295 407">Modifies how the field is written to XML. You can specify the following:</p> <p data-bbox="553 420 1422 516">Output type This option is available if you are modifying a simple field. It specifies whether the dataflow field should be written to an XML element or attribute.</p> <p data-bbox="743 529 1422 625">Element Select this to write the field's data to an XML element. Specify the element name you want to use in the Element name field.</p> <p data-bbox="743 638 1422 735">Attribute Writes the field's data to an attribute of the parent element. Specify the attribute name you want to use in the Attribute name field.</p> <p data-bbox="553 764 1422 856">Element name/Attribute name Specifies the name of the element or attribute to be written to the XML file. The default name is the dataflow field name.</p> <p data-bbox="553 877 1422 974">Change all children to This option is available if you are modifying a complex element. It specifies the type of XML you want the complex element to contain. One of the following:</p> <p data-bbox="743 987 1422 1113">No change The child types remain as they are currently defined, either element or attribute. You can specify the type for each field individually by selecting the field and clicking Modify.</p> <p data-bbox="743 1125 1422 1188">Elements All simple fields under the element are written as XML elements.</p> <p data-bbox="743 1201 1422 1264">Attributes All simple fields under the element are written as XML attributes.</p> <p data-bbox="553 1297 1422 1394">Namespace If you want to specify an XML namespace to use for the element or attribute, select it here. You can create namespaces on the Fields tab of the Write to XML stage.</p> <p data-bbox="553 1415 1422 1507">Include empty fields Check this box to include in the output file XML elements that have a null value or no data. If you do not check this box, empty elements will not be included in the output.</p> <p data-bbox="743 1520 1422 1612">For example, if you define an element named <code><City></code> but there is a record that does not have any data in the City field, the XML output will contain the following if you check Include empty fields:</p> <pre data-bbox="743 1625 1123 1659"><City xs:nil="true"></City></pre> <p data-bbox="743 1671 1422 1734">If you do not check this box the <code><City></code> element will not be written to the output file.</p> <p data-bbox="639 1759 1422 1875">Note: Dataflow field displays the field whose data will be written to the element or attribute. This is displayed so that if you change the element or attribute name to something different you can still see which field's data is contained in the element or attribute.</p>

Option Name	Description
Remove	Removes the selected field from the output. If you remove a list field all child fields are also removed. If you remove a child field, just the selected child field is removed from the list field.
Remove All	Removes all the fields from the output.
Move Up/Move Down	Reorders the selected field. Note that you cannot move simple elements into complex elements. If you want to modify the elements in a complex element, you must modify your dataflow's Aggregator stage to include the dataflow fields you want in the complex element. For more information, see Creating Complex XML from Flat Data on page 280.
Regenerate	Replaces the fields currently defined with the fields coming into Write to XML from the upstream channel.

Runtime Tab

Option Name	Description
Generate multiple files	Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum™ Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Management Console. Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.
File path field	Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. Note that only the simple type elements mapped directly to a root will be listed in the File path field . This field is only enabled if you select the Generate multiple files .

Option Name	Description
Generate schema at runtime	Select this option to generate an XSD at runtime and insert a <code>noNamespaceSchemaLocation</code> reference to schema in the XML file. The value of attribute <code>noNamespaceSchemaLocation</code> is XSD file name in the XML file. If you export the schema while editing a dataflow, there will be no reference to the XSD in the output XML file and the user would need to manually add in the reference to the XSD.
Schema path	Specifies the path to save the XSD file that contains the schema of the output XML file. Click the ellipses button (...) to browse to the file you want. The schema file is saved to the location you specify when you run the dataflow.

Using Namespaces in an XML Output File

Namespaces allow you to have duplicate element and attribute names in your output by assigning each element or attribute to an XML namespace.

1. In Enterprise Designer, open the dataflow.
2. Double-click the Write to XML stage on the canvas.
3. Click the **Fields** tab.
4. Define one or more namespaces:
 - a) In the **Prefix** column, enter the prefix you want to use to associate an element or attribute with the namespace.
 - b) In the **Namespace** column, specify the URL of the namespace.
 - c) Repeat to define as many namespaces as you want to use for the output XML file.
5. Associate one or more elements or attributes to the namespace.
 - a) On the **Fields** tab, select the element or attribute you want to associate with a namespace then click **Modify**, or create a new element or attribute by clicking **Add**.
 - b) In the **Namespace** field, choose the namespace prefix you want to associate with the element or attribute.
 - c) Click **OK**.

Creating Complex XML from Flat Data

Dataflows often produce records containing flat fields which get written to XML as a simple XML elements. If you want to organize flat fields into complex XML elements to produce hierarchical data, you can do so using one or more Aggregator stages.

For example, given this flat data where the first line is a header record:

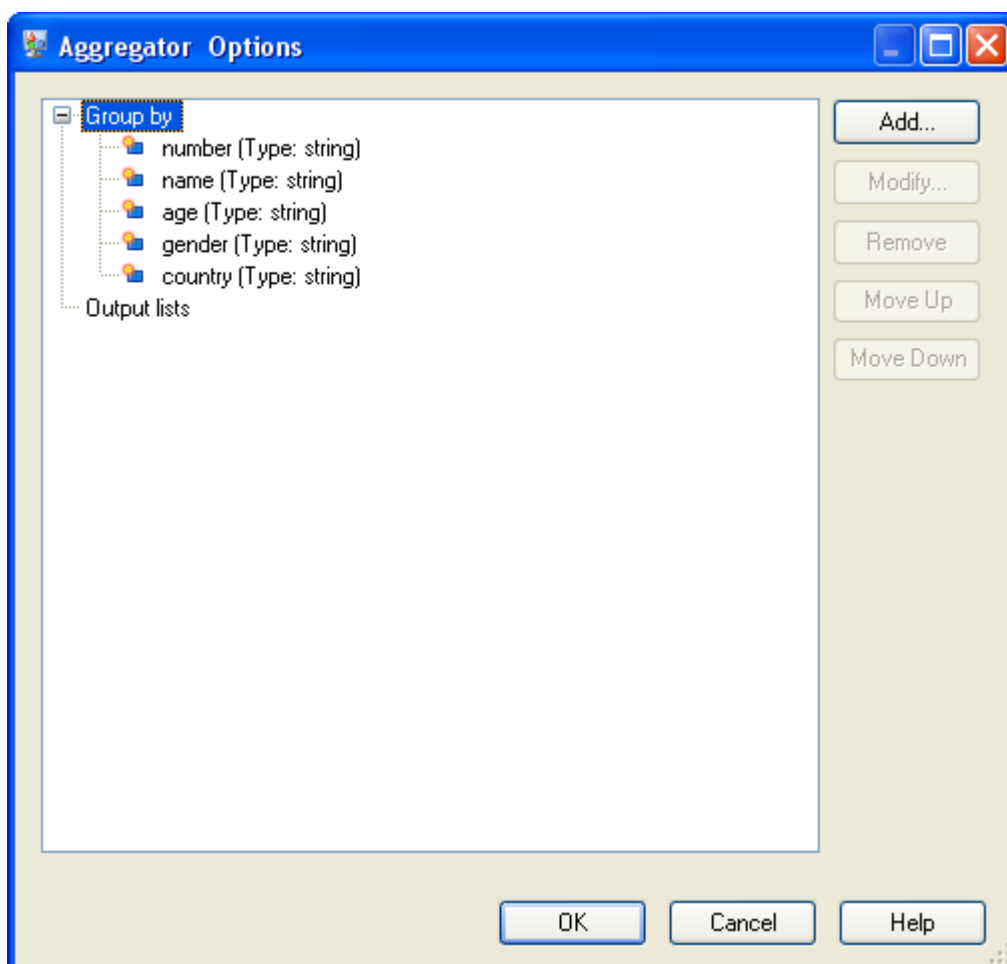
```
addressline1,age,city,country,gender,name,number,postalcode,stateprovince,type
1253 Summer St.,43,Boston,United States,M,Sam,019922,02110,MA,Savings
```

You might want to group the fields of data related to the address and fields related to the account into complex XML elements named `<Address>` and `<Account>` as shown here:

```
<CustomerRecord>
  <name>Sam</name>
  <age>43</age>
  <gender>M</gender>
  <country>United States</country>
  <Address>
    <addressline1>1253 Summer St.</addressline1>
    <city>Boston</city>
    <stateprovince>MA</stateprovince>
    <postalcode>02110</postalcode>
  </Address>
  <Account>
    <number>019922</number>
    <type>Savings</type>
  </Account>
</CustomerRecord>
```

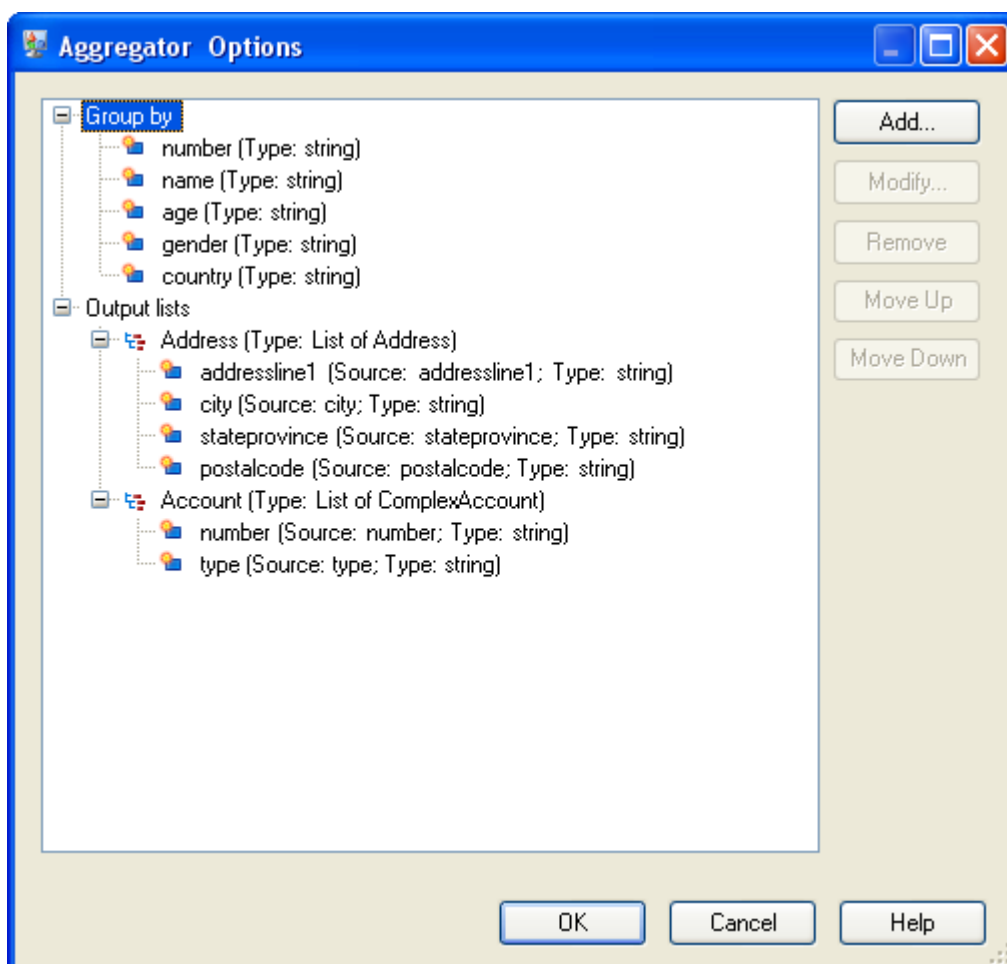
1. Add an Aggregator stage to the point in the dataflow where you want to construct complex elements.
2. Double-click the Aggregator stage to open the stage options.
3. Select **Group by** and click **Add**.
4. Select the field that contains a unique value for each record, such as an account number and click **OK**.
5. If there are other simple fields you want to pass through, select **Group by** and click **Add** again and add all the simple fields you want to include.

For example, in this case there are five simple fields that will be included in each record: number, name, age, gender, and country.



6. Select **Output lists** and click **Add**.
7. Select **New data type**. This will have the effect of defining a new complex element. Enter a description for the kind of data that this complex element will contain. For example, you could enter "Complex" since you are constructing a complex XML element. The data type name can be anything you want.
8. In the **Name** field, enter the name to use for the field. This will also be the name of the XML element.
9. Click **OK**.
10. Select the field you just created and click **Add**.
11. With **Existing field** selected, choose a field that you want to add as a child field to the complex element and click **OK**.
12. Repeat the previous two steps to add additional fields to the complex element.
13. Add additional complex fields as needed.

When you are finished, you should have an Aggregator stage that lists each simple and complex field you want to include in each record. For example:



14. Click **OK**.

Date and Number Patterns

Date and Time Patterns

When defining data type options for date and time data, you can create your own custom date or time pattern if the predefined ones do not meet your needs. To create a date or time pattern, use the notation described in the following table. For example, this pattern:

dd MMMM yyyy

Would produce a date like this:

14 December 2012

Letter	Description	Example
G	Era designator	AD
yy	Two-digit year	96
yyyy	Four-digit year	1996
M	Numeric month of the year.	7
MM	Numeric month of the year. If the number is less than 10 a zero is added to make it a two-digit number.	07
MMM	Short name of the month	Jul
MMMM	Long name of the month	July
w	Week of the year	27
ww	Two-digit week of the year. If the week is less than 10 an extra zero is added.	06
W	Week of the month	2
D	Day of the year	189
DDD	Three-digit day of the year. If the number contains less than three digits, zeros are added.	006
d	Day of the month	10
dd	Two-digit day of the month. Numbers less than 10 have a zero added.	09
F	Day of the week in month	2
E	Short name of the day of the week	Tue
EEEE	Long name of the day of the week	Tuesday

Letter	Description	Example
a	AM/PM marker	PM
H	Hour of the day, with the first hour being 0 and the last hour being 23.	0
HH	Two-digit hour of the day, with the first hour being 0 and the last hour being 23. Numbers less than 10 have a zero added.	08
k	Hour of the day, with the first hour being 1 and the last hour being 24.	24
kk	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
K	Hour hour of the morning (AM) or afternoon (PM), with 0 being the first hour and 11 being the last hour.	0
KK	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
h	Hour of the morning (AM) or afternoon (PM), with 1 being the first hour and 12 being the last hour.	12
hh	Two-digit hour of the morning (AM) or afternoon (PM), with 1 being the first hour and 12 being the last hour. Numbers less than 10 have a zero added.	09
m	Minute of the hour	30
mm	Two-digit minutes of the hour. Numbers less than 10 have a zero added.	05
s	Second of the minute	55
ss	Two-digit second of the minute. Numbers less than 10 have a zero added.	02
S	Millisecond of the second	978

Letter	Description	Example
SSS	Three-digit millisecond of the second. Numbers containing fewer than three digits will have one or two zeros added to make them three digits.	978 078 008
z	Time abbreviation of the time zone name. If the time zone does not have a name, the GMT offset.	PST GMT-08:00
zzzz	The full time zone name. If the time zone does not have a name, the GMT offset.	Pacific Standard Time GMT-08:00
Z	The RFC 822 time zone.	-0800
X	The ISO 8601 time zone.	-08Z
XX	The ISO 8601 time zone with minutes.	-0800Z
XXX	The ISO 8601 time zone with minutes and a colon separator between hours and minutes.	-08:00Z

Number Patterns

When defining data type options for numeric data, you can create your own custom number pattern if the predefined ones do not meet your needs. A basic number pattern consists of the following elements:

- A prefix such as a currency symbol (optional)
- A pattern of numbers containing an optional grouping character (for example a comma as a thousands separator)
- A suffix (optional)

For example, this pattern:

```
$ ###,###.00
```

Would produce a number formatted like this (note the use of a thousands separator after the first three digits):

```
$232,998.60
```

Patterns for Negative Numbers

By default, negative numbers are formatted the same as positive numbers but have the negative sign added as a prefix. The character used for the number sign is based on the locale. The negative sign is "-" in most locales. For example, if you specify this number pattern:

```
0.00
```

The number negative ten would be formatted like this in most locales:

```
-10.00
```

However, if you want to define a different prefix or suffix to use for negative numbers, specify a second pattern, separating it from the first pattern with a semicolon (";"). For example:

```
0.00; (0.00)
```

In this pattern, negative numbers would be contained in parentheses:

```
(10.00)
```

Scientific Notation

If you want to format a number into scientific notation, use the character `E` followed by the minimum number of digits you want to include in the exponent. For example, given this pattern:

```
0.###E0
```

The number 1234 would be formatted like this:

```
1.234E3
```

In other words, 1.234×10^3 .

Note the following:

- The number of digit characters after the exponent character gives the minimum exponent digit count. There is no maximum.
- Negative exponents are formatted using the localized minus sign, not the prefix and suffix from the pattern.
- Scientific notation patterns cannot contain grouping separators (for example, a thousands separator).

Special Number Pattern Characters

The following characters are used to produce other characters, as opposed to being reproduced literally in the resulting number. If you want to use any of these special characters as literal characters in your number pattern's prefix or suffix, surround the special character with quotes.

Symbol	Description
0	<p>Represents a digit in the pattern including zeros where needed to fill in the pattern. For example, the number twenty-seven when applied to this pattern:</p> <p>0000</p> <p>Would be:</p> <p>0027</p>
#	<p>Represents a digit but zeros are omitted. For example, the number twenty-seven when applied to this pattern:</p> <p>####</p> <p>Would be:</p> <p>27</p>
.	<p>The decimal separator or monetary decimal separator used in the selected locale. For example, in the U.S. the dot (.) is used as the decimal separator but in France the comma (,) is used as the decimal separator.</p>
-	<p>The negative sign used in the selected locale. For most locals this is the minus sign (-).</p>
,	<p>The grouping character used in the selected locale. The appropriate character for the selected locale will be used. For example, in the U.S., the comma (,) is used as a separator.</p> <p>The grouping separator is commonly used for thousands, but in some countries it separates ten-thousands. The grouping size is a constant number of digits between the grouping characters, such as 3 for 100,000,000 or 4 for 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. For example, all the following patterns produce the same result:</p> <p>#, ##, ###, ####</p> <p>##### , #####</p> <p>##, ####, #####</p>
E	<p>Separates mantissa and exponent in scientific notation. You do not need to surround the E with quotes in your pattern. See Scientific Notation on page 287.</p>
;	<p>Separates positive and negative subpatterns. See Patterns for Negative Numbers on page 287.</p>

Symbol	Description
<code>%</code>	<p>Multiply the number by 100 and show the number as a percentage. For example, the number .35 when applied to this pattern:</p> <pre>##%</pre> <p>Would produce this result:</p> <pre>35%</pre>
<code>¤</code>	<p>The currency symbol for the selected locale. If doubled, the international currency symbol is used. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.</p>
<code>'</code>	<p>Used to quote special characters in a prefix or suffix. For example:</p> <pre>''#'#"</pre> <p>Formats 123 to:</p> <pre>"#123"</pre> <p>To create a single quote itself, use two in a row:</p> <pre>"# o'clock"</pre>

6 - Configurations

In this section

Oracle LogMiner Configurations

291

Oracle LogMiner Configurations

Oracle LogMiner is a backend utility that allows Spectrum to query and access the logs created for an Oracle database.

The utility allows Spectrum™ to read the Oracle data source's logs to track the changes made to columns of its tables, as part of the **DB Change Data Reader** stage.

Unsupported Datatypes and Table Storage Attributes

Oracle LogMiner does not support the following datatypes and table storage attributes:

- **BFILE** datatype
- Simple and nested abstract datatypes (**ADTs**)
- Collections (nested tables and VARRAYs)
- Object references
- Tables on which compression has been enabled
- SecureFiles

Supported Databases and Redo Log File Versions

LogMiner runs on Oracle databases version 8.1 or later.

You may use LogMiner to analyze redo log files from Oracle 8.0 databases as well. However, the information retrieved depends on the version of the log and not the version of the database in use.

For example, to use LogMiner optimally, the redo log files for Oracle9i can be augmented to capture additional information when supplemental logging is enabled. Redo log files created with older versions of Oracle do not have the additional data and may therefore have limitations on the operations and datatypes supported by LogMiner.

*SQL*Loader Restrictions*

Spectrum CDC can capture data that was loaded into Oracle tables by the SQL*Loader utility. However, the following restrictions apply:

1. The data load must be through *conventional path*. Spectrum CDC cannot capture data that is loaded by a *direct path load* as Oracle LogMiner does not support direct path loads.
2. The load method should be INSERT, APPEND, or REPLACE.

TRUNCATE is not supported as the TRUNCATE command causes the SQL*Loader to issue the TRUNCATE TABLE DDL command. Since the Spectrum CDC feature does not capture the mentioned DDL, row deletions that result from using the TRUNCATE TABLE DDL command are not captured.

Required User Privileges

The following table identifies the minimum system privileges that Oracle CDC users must have:

System Privilege	Oracle Version
ALTER ANY TABLE	ALL
CONNECT	ALL
LOCK ANY TABLE	ALL
SELECT ANY TRANSACTION	10g or later

The following table identifies the minimum object privileges that Oracle CDC users must have:

Object Name	Privilege
Source Tables	LOCK ANY TABLE OR SELECT
PUBLIC.V\$DATABASE	SELECT
PUBLIC.V\$LOGMNR_CONTENTS	SELECT
SYS.DBMS_LOGMNR	EXECUTE
SYS.DBMS_LOGMNR_D	EXECUTE

For more information on Oracle LogMiner, see [here](#).

7 - Optimizing Performance

In this section

Determining an Optimum Fetch Size

294

Determining an Optimum Fetch Size

In the **Read from DB** stage, the optimum fetch size is calculated by taking execution time readings between a **Read from DB** stage and a **Write to Null** stage.

Ensure you test execution times of the test job with the different fetch size values using your own application with the Spectrum™ Data Integration Module.

1. Create a job in the Enterprise Designer.
2. Drag a **Read from DB** stage onto the canvas.
3. Drag a **Write to Null** stage onto the canvas.
4. Create a channel between the two stages.
5. Double-click on the **Read from DB** stage to configure it to read data from a table containing the test data.
 - a) In the **General** tab, in the **Connection** field, select the database that contains the test data.
 - b) Click **Build SQL...** to create the SQL query using selected schema and tables from which to read the data.

Ensure the selected table has at least a 1000 records to allow optimum test values.
 - c) In the **Runtime** tab, check the **Fetch size** checkbox.
 - d) In the attached field, enter the number of records you want to read in one instance.

The Spectrum™ Technology Platform has been tested to work optimally with a Fetch size of up to 1000.
 - e) Click **OK**.
6. Save the job.
7. Run the job.

The **Execution Details** window opens.
8. Click **Refresh**.
9. Note the **Started** and **Finished** times.
10. Repeat the steps 7 - 9, gradually increasing the fetch size to identify the optimal setting for your server.

You now have identified the fetch size setting that provides the optimal performance for your environment.

Notices

© 2017 Pitney Bowes Software Inc. All rights reserved. MapInfo and Group 1 Software are trademarks of Pitney Bowes Software Inc. All other marks and trademarks are property of their respective holders.

USPS® Notices

Pitney Bowes Inc. holds a non-exclusive license to publish and sell ZIP + 4® databases on optical and magnetic media. The following trademarks are owned by the United States Postal Service: CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS^{Link}, NCOA^{Link}, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite^{Link}, United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code, and ZIP + 4. This list is not exhaustive of the trademarks belonging to the Postal Service.

Pitney Bowes Inc. is a non-exclusive licensee of USPS® for NCOA^{Link}® processing.

Prices for Pitney Bowes Software's products, options, and services are not established, controlled, or approved by USPS® or United States Government. When utilizing RDI™ data to determine parcel-shipping costs, the business decision on which parcel delivery company to use is not made by the USPS® or United States Government.

Data Provider and Related Notices

Data Products contained on this media and used within Pitney Bowes Software applications are protected by various trademarks and by one or more of the following copyrights:

© Copyright United States Postal Service. All rights reserved.
 © 2014 TomTom. All rights reserved. TomTom and the TomTom logo are registered trademarks of TomTom N.V.

© 2016 HERE

Fuente: INEGI (Instituto Nacional de Estadística y Geografía)

Based upon electronic data © National Land Survey Sweden.

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

Portions of this program are © Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Second Decimal, LLC

© Copyright Canada Post Corporation

This CD-ROM contains data from a compilation in which Canada Post Corporation is the copyright owner.

© 2007 Claritas, Inc.

The Geocode Address World data set contains data licensed from the GeoNames Project (www.geonames.org) provided under the Creative Commons Attribution License ("Attribution

License") located at <http://creativecommons.org/licenses/by/3.0/legalcode>. Your use of the GeoNames data (described in the Spectrum™ Technology Platform User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with Pitney Bowes Software, Inc. and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.



3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com