IBM InfoSphere Information Server
Version 11 Release 3

Guide for the Dynamic RDBMS Stage

SC19-4348-00

IBM InfoSphere Information Server
Version 11 Release 3

Guide for the Dynamic RDBMS Stage

SC19-4348-00

Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 73.

Contents

Dynamic RDBMS stage

Use the Dynamic RDBMS stage in IBM® InfoSphere® DataStage® parallel and server jobs to access various types of relational databases.

When you use IBM InfoSphere DataStage to access various types of relational databases, you can choose from a collection of connectivity options. For new jobs, use the DRS Connector stage, which offers better functionality and performance than the Dynamic RDBMS stage.

If you have jobs that use the Dynamic RDBMS stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Overview

Use the Dynamic RDBMS stage to access relational database management systems by using the native interfaces that are available for the corresponding databases. The choice of the database type is not determined when the stage is placed on the job canvas but is instead specified when the stage is configured.

In addition, an InfoSphere DataStage job parameter can be used for the database type setting in the stage and in that case the selection of the database type is made when the job runs.

The Dynamic RDBMS stage similar to ODBC connectivity stages in that a single stage can be configured to access many different database types. An important difference is that the Dynamic RDBMS accesses databases directly through the native database client libraries, while ODBC Stages access databases through the ODBC drivers.

The Dynamic RDBMS stage is supported primarily for backward compatibility so that jobs developed in previous InfoSphere DataStage versions can continue to run after they have been imported into the new environment. This stage supports IBM DB2®, Informix®, Microsoft SQL Server, Oracle, Sybase and ODBC data sources.

The Dynamic RDBMS stage supports the following features:

- Database read, write and lookup operations.
- Automatic generation of SQL statements at runtime based on the specified table name, column definitions on the link and other user defined properties of the stage. The SQL statements can also be manually specified.
- Running an SQL statement specified in a file that resides on the engine tier.
- Automatically creating or replacing target tables in the database during the job runtime.
- Automatically clearing the contents of the existing target table at runtime before writing data to it.
- Writing rows to the database and reading rows from the database in batch (array) mode using an array of the specified size.
- Specifying a transaction size (number of records per transaction).
- Specifying an isolation level for the transactions.

- Writing data to the database in bulk load mode using native bulk load interfaces provided by the database.
- Rejecting bad records (server jobs only). Requires the use of the Transformer stage.
- Record ordering across multiple input links.
- Running user-specified SQL statements on the database after the job starts and before any records are actually processed by the stage. The statements can be specified directly in the stage or in a file that resides on the engine tier.
- Running user-specified SQL statements on the database after all the records have been processed by the stage and before the job ends. The statements can be specified directly in the stage or in a file that resides on the engine tier.
- Derivation attributes for the columns on the output link of the stage. The stage uses derivation values instead of column names when it generates `SELECT` statements to use at runtime for reading data from the database.
- Handling the specified table name and column names as case-sensitive values or case-insensitive values.

Configuration

To connect to the database server when the job runs, the stage needs to be able to locate and load database client libraries installed on the engine tier computer. You must set the library path environment variable to include the directory where the database client libraries reside.

Setting environment variables

You can configure the environment variables at the operating system level so that they automatically apply to the operating system user who is running the jobs on the computer that you installed the InfoSphere Information Server engine.

About this task

On Linux or UNIX operating systems the environment variables can also be specified in the dsenv script located in the *$ISHOME*`/Server/DSEngine` directory, where *$ISHOME* is the InfoSphere Information Server home directory (/opt/IBM/InformationServer by default). When environment variables are specified in the dsenv script, they automatically apply to all InfoSphere DataStage projects running under that engine.

Any time a change is made to the environment variables in the dsenv script, the InfoSphere Information Server engine and ASB Agent processes need to be restarted in order to detect the change. The following steps for restarting these processes assume that you installed InfoSphere Information Server in the /opt/IBM/InformationServer directory. If InfoSphere Information Server is installed in another location, modify the commands accordingly.

Procedure

1. Login to the computer that you installed the Engine tier on as the root user.

   Alternatively, if the currently logged in user has sudo privileges granted, issue the following command to gain root access:

   sudo su
2. Change directory to the InfoSphere Information Server engine home directory:

   cd /opt/IBM/InformationServer/Server/DSEngine

3. Source the dsenv script (note the two period characters in this command):
   . ./dsenv
4. Stop the InfoSphere Information Server engine:
   bin/uv –admin –stop
5. Once the InfoSphere Information Server engine has stopped, start it again:
   bin/uv –admin –start
6. Change directory to the ASB Agent home directory:
   cd /opt/IBM/InformationServer/ASBNode/bin
7. Stop the ASB Agent processes:
   ./NodeAgents.sh stopAgent
8. Once the ASB Agent has stopped, start it again:
   ./NodeAgents.sh start

Results

Once the ASB Agent process has been restarted it can take a little time before the Information Server domain services (WebSphere® Application Server) have registered the event (typically a minute or so). This applies also in the case when the services tier and the engine tier are residing on the same machine.

Information Server installations on Windows operating system do not include dsenv script. In order for InfoSphere Information Server and ASB Agent processes to detect changes in the environment variables, the environment variable changes need to be made at the system level because these processes are running under the built-in Local System account. After making the changes it is necessary to restart the DataStage Server machine.

Environment variables can also be configured for individual DataStage projects or jobs. This can be done in InfoSphere Information Server installations on Linux or UNIX and Windows operating systems. For an environment variable to apply to all jobs in a project, it is necessary to define the environment variable at the project level through the InfoSphere DataStage and QualityStage® Administrator. Then if necessary to use a different value for the environment variable in a particular job, add the environment variable to the job as a job parameter and then set its value at the job level through the **Edit** > **Job Properties** main menu option in DataStage Designer.

Note that the library path and path environment variables are predefined environment variables so they cannot be added to the project as user-defined environment variables. Further note that on the Windows operating system both library path and path are represented by the same PATH environment variable.

It is important to understand that the values specified for the library path and path environment variables at the project or job level are appended to the existing system values for these variables. For example if directory `A` is specified as the value for the library path environment variable at the project level, and directory `B` with the same library names was already present in the library path defined at the operating system level or the dsenv script level, the libraries from the conflicting directory `B` will be loaded at runtime because this directory will appear before directory `A` in the effective library path value at runtime.

Configuration for IBM DB2

The IBM DB2 client software must be installed and configured on the computer that you installed the InfoSphere Information Server engine.

The DB2 database server needs to be accessible from the DB2 client. The database server node and the database need to be properly cataloged in the database directory on the DB2 client side. You should test the connectivity to the DB2 database outside of InfoSphere DataStage and ensure that it works properly. For example, you can use the DB2 Command Line Processor tool to verify this connection. Refer to the IBM DB2 product documentation for details on how to install and configure the DB2 client and server software.

The library path environment variable needs to include the DB2 client library directory. The default DB2 client library directories are:

Linux or UNIX
`/opt/IBM/db2/V9/lib64`

Microsoft Windows
`C:\IBM\SQLLIB\BIN`

The *DB2INSTANCE* environment variable needs to be set to the name of the DB2 instance to be used as the active DB2 instance at job runtime. The *DB2INSTANCE* environment variable needs to be set even if the stage is accessing default DB2 instance. The default values are:

Linux or UNIX
`db2inst1`

Microsoft Windows
`DB2`

The *DB2CODEPAGE* environment variable needs to be set if the NLS map name for the job does not match the current system locale on the engine tier machine.

As an example consider a parallel job that reads data from a file encoded in Latin-1 (ISO-8859-1) character set encoding and writes data to a Dynamic RDBMS stage configured for IBM DB2 database type. In this example the column on the link between the Sequential File stage and the Dynamic RDBMS stage is defined as a VarChar column, and the Extended attribute for the column is not set. The system locale (*LANG* environment variable) is set to `en_US.UTF-8`. The *NLS map name* for the job is set to `ISO-8859-1:1987`. In this case the Dynamic RDBMS stage will be passing data to DB2 client encoded using NLS map name (ISO-8859-1), but the DB2 client will assume that the data is encoded according to the system locale character set (UTF-8). Therefore data corruption can occur with invalid data inserted to the database. To notify the DB2 client of the actual encoding of the data, the *DB2CODEPAGE* environment variable would in this case need to be set to the Coded Character Set Identifier (CCSID) value of the character set in which the data is encoded. In this case (ISO-8859-1 encoding) the corresponding CCSID value is 819.

As another example, the *CCSID* value of the UTF-8 character set encoding is `1208`.

Note: The *DB2CODEPAGE* can also be set as the DB2 registry variable.

For additional information about the configuration requirements for DB2 connectivity in InfoSphere DataStage in general, refer to the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for IBM DB2 Databases*.

Configuration for ODBC

The library path environment variable needs to include the directory with the ODBC driver manager and driver libraries.

On the Windows operating system, the ODBC driver manager library is provided by the operating system and its directory is automatically included in the system library path environment variable (*PATH*). On UNIX or Linux operating systems the ODBC driver manager is included with InfoSphere Information Server.

InfoSphere Information Server includes a set of ODBC driver libraries and the driver manager library on UNIX or Linux. The ODBC directory with these libraries is automatically added to the library path during the InfoSphere Information Server installation. On Windows this directory is added to the system library path (*PATH*) and on UNIX or Linux it is added to the dsenv InfoSphere DataStage script. The location of the ODBC directory depends on the operating system and the InfoSphere Information Server home directory. By default the location of the ODBC directories are:

UNIX or Linux

```
/opt/IBM/InformationServer/Server/branded_odbc/lib
```

Microsoft Windows

```
C:\IBM\InformationServer\ODBCDrivers
```

On UNIX or Linux platforms the ODBCINI environment variable needs to be set to point to the .odbc.ini file which contains the ODBC data source name (DSN) definitions. This environment variable is set in the dsenv InfoSphere DataStage script automatically as part of the InfoSphere Information Server installation process. By default, the values are:

UNIX or Linux

```
/opt/IBM/InformationServer/Server/DSEngine/.odbc.ini
```

Microsoft Windows

The ODBCINI environment variable is not applicable since the data source definitions are managed by the ODBC driver manager application included with the operating system.

The ODBC DSN definitions need to be configured as System DSN definitions in the ODBC Data Source Administrator.

For additional information about the configuration requirements for ODBC connectivity in InfoSphere DataStage in general and how to configure ODBC DSN definitions refer to the *IBM InfoSphere Information Server Planning, Installation, and Configuration Guide*.

Configuration for Oracle

The Oracle database must be accessible from the Oracle client.

The Oracle database needs to be accessible from the Oracle client and the connectivity between Oracle client and Oracle database server needs to be tested, for example using the Oracle SQL*Plus utility. There are several methods for configuring access to an Oracle database from the Oracle client machine and these

steps typically involve updating Oracle configuration files such as tnsnames.ora and sqlnet.ora. Refer to the Oracle product documentation for details on how to perform those steps.

The library path environment variable needs to include the directory with Oracle client libraries. For example, the Oracle client library directories could be:

Linux or UNIX

```
/u01/app/oracle/product/11.2.0/client_1/lib
```

Microsoft Windows

```
C:\app\username\product\11.2.0\client_1\BIN
```

where *username* represents a local operating system user name.

If the complete Oracle database product is installed on the InfoSphere Information Server engine computer instead of just the Oracle client product, then instead of `client_1` the path will typically include `dbhome_1`.

Dynamic RDBMS stage does not support Oracle Instant Client installations. Only the standard Oracle Client installation is supported.

Either the *ORACLE_HOME* or *TNS_ADMIN* environment variable needs to be set in order for the stage (and the Oracle client libraries) to be able to access Oracle configuration file tnsnames.ora. If *ORACLE_HOME* is specified, then the tnsnames.ora file needs to reside under *$ORACLE_HOME*`/network/admin` directory. If *TNS_ADMIN* environment variable is specified, then the `tnsnames.ora` file needs to reside directly under the *$TNS_ADMIN* directory. If both environment variables are specified, the *TNS_ADMIN* environment variable takes precedence.

It is recommended to set *NLS_LANG* environment variable as well. This environment variable specifies the language, country and character set settings for the Oracle client libraries. The value should be compatible with the current operating system locale settings on the InfoSphere Information Server engine computer and also with the NLS map name specified for the project. For example, on the InfoSphere Information Server engine computer with the en_US.UTF-8 locale settings and the InfoSphere DataStage NLS map name in the project set to value UTF-8, the *NLS_LANG* environment variable for the project would typically need to be set to the following value: `AMERICAN_AMERICA.AL32UTF8`.

If the character set in the *NLS_LANG* environment variable does not match the character set of the NLS map name setting, the Oracle client will assume that the data exchanged with the stage is encoded according to the *NLS_LANG* setting when it can be actually encoded according to the NLS map name. This can lead to data corruption and invalid values stored to the database or retrieved from the database. Therefore it is important to synchronize the *NLS_LANG* environment variable and the NLS map name values used for the job.

If the *NLS_LANG* environment variable is not set, it defaults to the following value: `AMERICAN_AMERICA.US7ASCII`. On Microsoft Windows installations the *NLS_LANG* value can also be set in the **Windows Registry**. To determine the effective NLS_LANG value used by the stage at runtime, design a job that uses the Dynamic RDBMS stage, add the CC_MSG_LEVEL environment variable to the project and add it to the job as a job parameter. Set it to value 2 and run the job. The job log will report the effective NLS_LANG value for the stage.

For additional information about the configuration requirements for Oracle connectivity in InfoSphere DataStage in general, refer to the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for Oracle Databases*.

Configuration for Informix

The Informix database must be accessible from the Informix client.

You must define the *INFORMIXDIR* environment variable ensure that it points to the IBM Informix product home directory.

Typical *INFORMIXDIR* environment variable settings are:

Linux or UNIX

`/usr/informix`

The library path environment variable needs to include $*INFORMIXDIR*`/lib directory` and the path environment variable needs to include $*INFORMIXDIR*/bin directory

The *ODBCINI* environment variable needs to point to the `.odbc.ini` file in which Informix connection definitions are created. This environment variable is set automatically during the InfoSphere Information Server installation.

Microsoft Windows

`C:\Program Files\IBM\Informix\Connect\`

The path environment variable needs to include `%`*INFORMIXDIR*`%\bin` directory.

The Informix client product includes a utility called **setnet32** which can be used to set *INFORMIXDIR* and other Informix specific environment variables. Refer to the Informix product documentation for more information about this utility and Informix environment variables in general.

The *INFORMIXSERVER* environment variable can be used to specify the name of the Informix server that you want to connect to. The Informix server name can also be specified explicitly in the connection definition when the stage is configured.

For additional information about the configuration requirements for Informix connectivity in InfoSphere DataStage in general, and for examples of DSN definitions in the `.odbc.ini` file, refer to the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for IBM Informix Databases*.

Configuration for Microsoft SQL Server

To configure Microsoft SQL Server, the Microsoft SQL Server database needs to be accessible from the Microsoft SQL Server client and the connectivity between Microsoft SQL Server client and Microsoft SQL Server database server needs to be tested.

You can test the connection, for example, using the SQL Server Management Studio tool in SQL Server 2005 and later, or the Query Analyzer tool in earlier SQL Server versions. When connecting to a remote database, ensure that the database server is configured to allow remote connections over the TCP/IP protocol.

On UNIX or Linux platforms, the *ODBCINI* environment variable needs to point to the `.odbc.ini` file in which Microsoft SQL Server connection definitions will be created.

Note: The Microsoft SQL Server Client software installation does not apply to Linux and UNIX platforms, therefore the Dynamic RDBMS stage on these platforms does not support Bulk insert mode of operation when configured for Microsoft SQL Server database type.

For additional information about the configuration requirements for SQL Server connectivity in InfoSphere DataStage in general, refer to the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for Microsoft SQL Server and OLE DB Data*.

Configuration for Sybase

The Sybase client software, such as Sybase Open Client, must be installed on the computer that you installed the InfoSphere Information Server engine on.

The Sybase database server needs to be accessible from the Sybase client. The database needs to be properly registered on the Sybase client side. Sybase Open Client provides a utility called dsedit which can be used to configure connection to the Sybase database.

The connectivity to Sybase database should be tested outside of InfoSphere DataStage and ensured that it works properly. For example, the isql tool in Sybase Open Client can be used to verify this connection. Refer to the Sybase product documentation for details on how to install and configure Sybase client and server software.

The following is a brief overview of the steps that typically need to be configured for the Sybase Open Client environment on the InfoSphere Information Server engine computer. For more details and for the information specific to your environment refer to the Sybase product installation and configuration documentation.

- The *SYBASE* environment variable needs to point to the Sybase product installation home directory. For example, typical values are:

  Linux or Unix
  `/opt/sybase`

  Microsoft Windows
  `C:\SYBASE`

- The *SYBASE_OCS* environment variable needs to point to the name of the Sybase Open Client directory. This value indicates the version and release of the Open Client product. For example this value can be OCS-12_5 or OCS-15_0.
- The library path environment variable must include:

  Linux or Unix
  the `$SYBASE/$SYBASE_OCS/lib` directory.

  Microsoft Windows
  `%SYBASE%\%SYBASE_OCS%\bin` and `%SYBASE%\%SYBASE_OCS%\dll` directories,

  where *SYBASE* and *SYBASE_OCS* represent the Sybase product installation home directory and Sybase Open Client directory.
- The *DSQUERY* environment variable can also be set. When set it specifies the name of the Sybase database server to connect to by default when the server

name is not provided explicitly in the connection request. The server with the name specified in this this environment variable should be defined in the local configuration files (such as `sql.ini` interfaces file) or the external directory service if it is configured. If the environment variable is not set, the default value `SYBASE` is used.

Settings for the Dynamic RDBMS stage

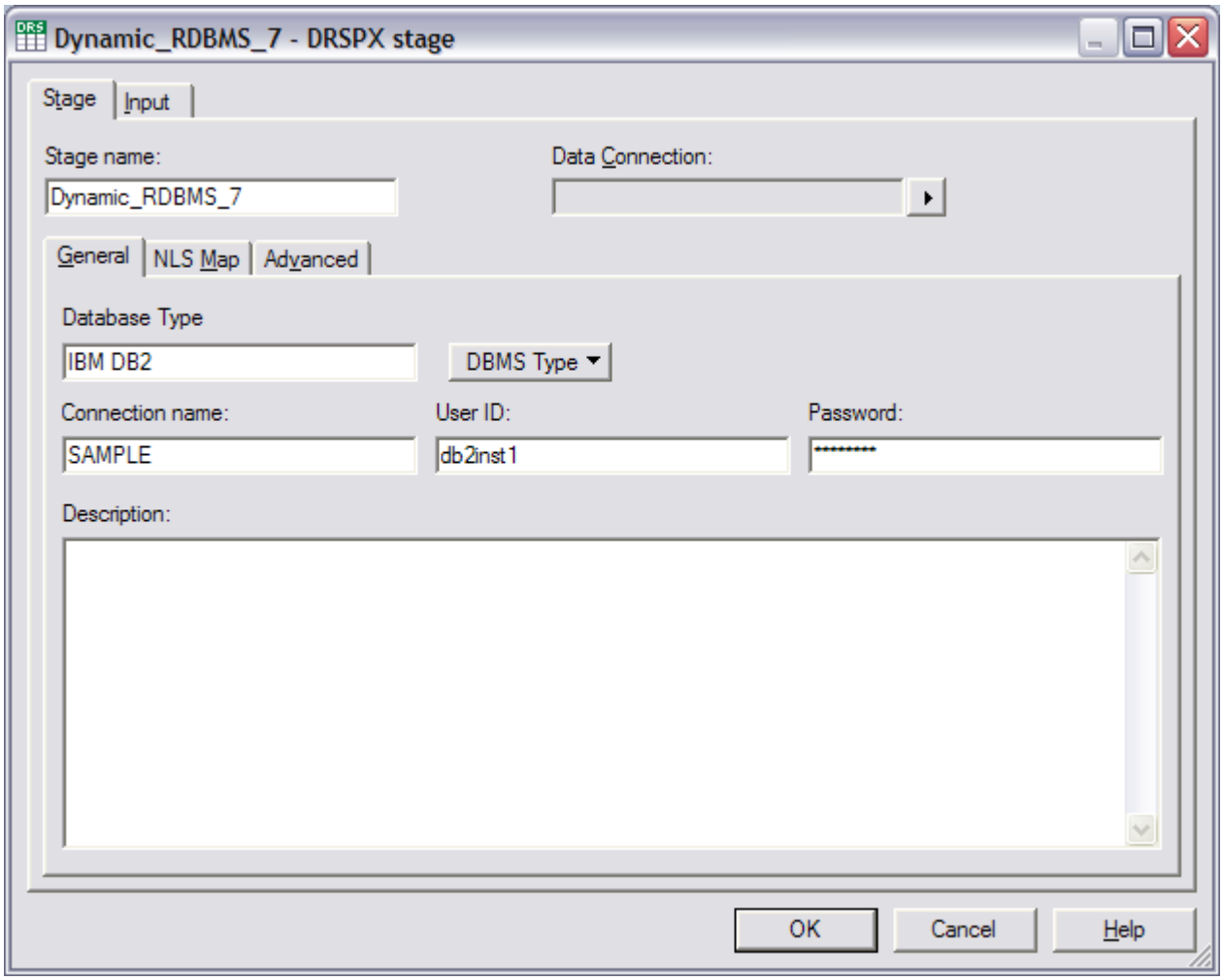
After the Dynamic RDBMS stage is placed on the job canvas, it needs to be configured to perform the operation intended by the job design.

Double-click the stage to open the **Stage** dialog box. You can set the properties on the **Stage** dialog box to the appropriate values depending on what action is expected to be performed by the stage when the job runs.

Dynamic RDBMS stage settings

The Dynamic RDBMS stage dialog box is organized into pages. The pages contain tabs, which in turn can contain additional tabs. The properties are located on the tabs.

The following figure shows the Dynamic RDBMS stage dialog box with the **Stage** page and **General** tab selected.



Stage name
Shows the name of the current stage instance. The name can be changed. In this example, the Stage name is set to `Dynamic_RDBMS_7`.

Data Connection
Load a previously saved Dynamic RDBMS connection definition from the repository, or to save the current connection definition to the repository so that it can be reused in other Dynamic RDBMS stage instances.

General
Shows stage level properties. On this tab, you select the database type and specifies connection details.

Description
Provides a description of the current stage instance.

NLS Map
Specifies the InfoSphere DataStage National Language Support (NLS) map name for the stage. It can be used to override the NLS map name settings specified at the job level.

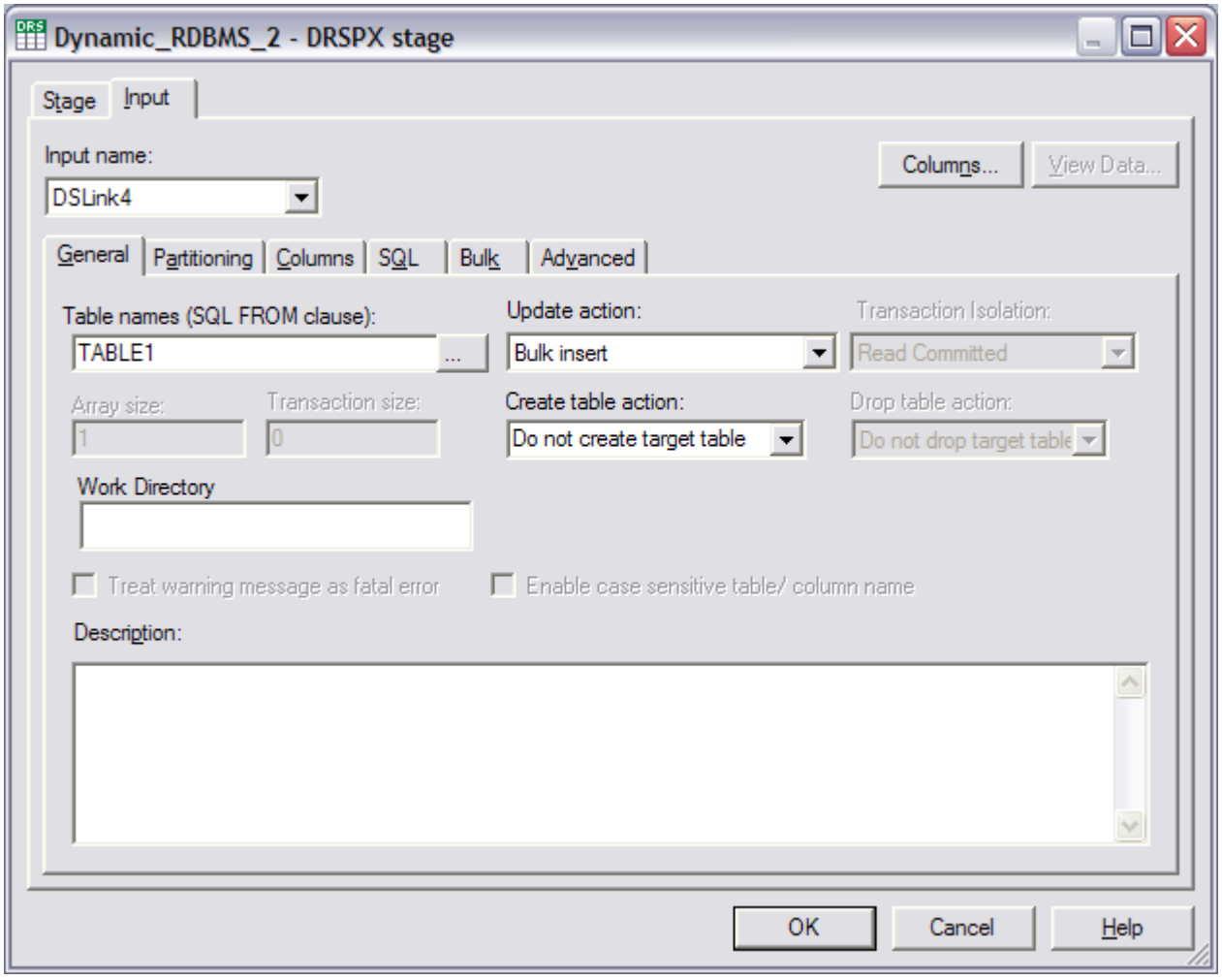
Advanced
Displayed only for stages on the parallel job canvas. It provides settings to choose whether to run the stage in sequential or parallel mode. If the stage is running in parallel this tab also provides option to constrain the stage to

run on a specific subset of processing nodes from the parallel engine configuration file (`default.apt` file by default).

Dynamic RDBMS link settings

When you select a link from the **Input name** list, the settings for that link are shown.

The following figure shows where settings for the selected link `DSLink4` are displayed.



General
Shows the properties that can be defined at the link level. The figure shows setting for input type of link. Output and reference links have different properties but their organization is similar.

Description
Specify a description for the currently selected link.

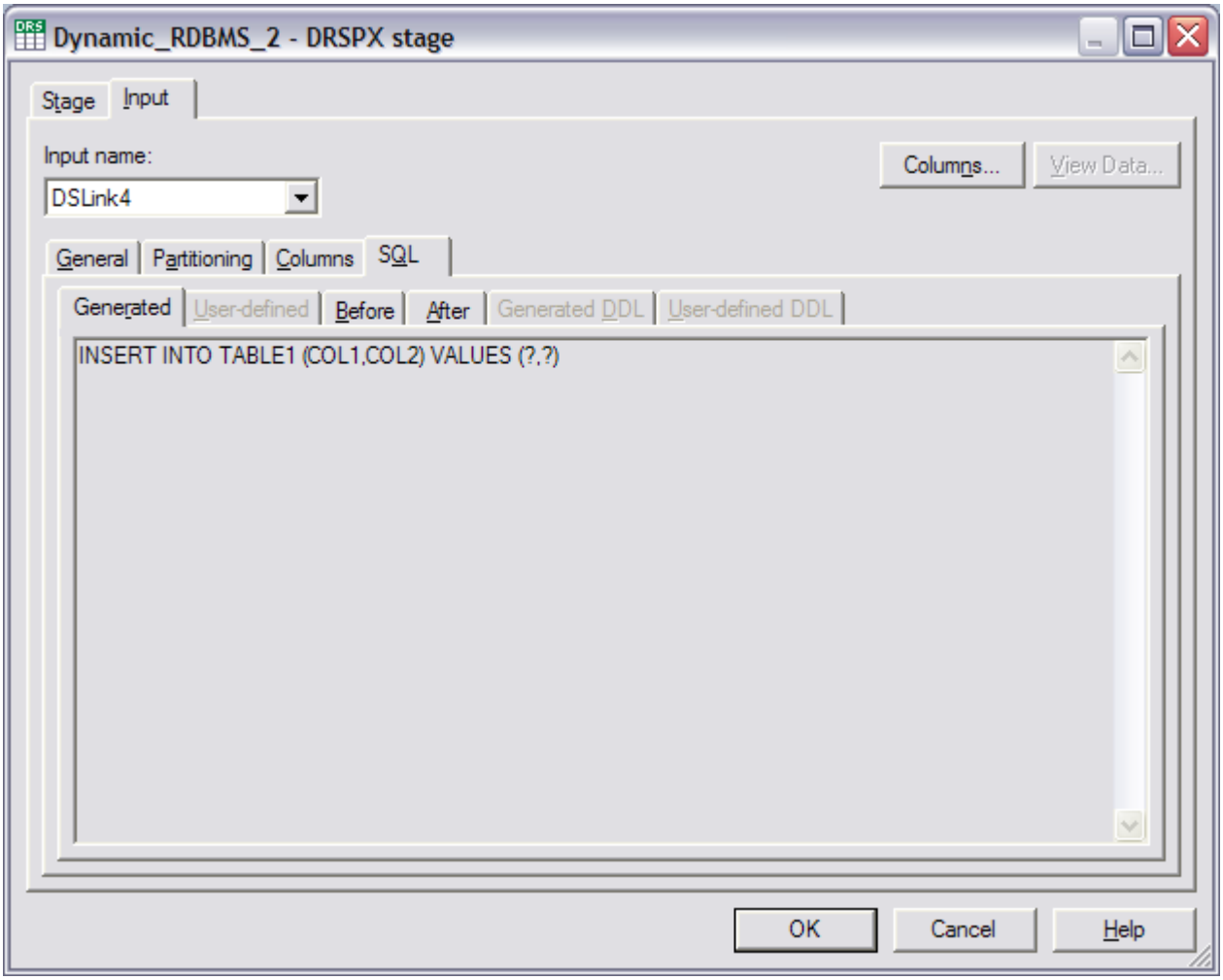
Partitioning
Displayed only for input links of the stages on the parallel canvas. Specify how records are partitioned across processing nodes on which the stage runs when configured to run in parallel mode.

Columns
Define columns on the link. The columns can be defined manually, or a table definition from the repository can be applied on the link.

Bulk Specify the settings for bulk load write mode of operation. It displays properties in a table format.

Advanced
Displayed only for the stages on the parallel canvas. Specify buffering strategy for records.

SQL Contains additional tabs as shown in the following figure.



Generated
Displays the read-only SQL statements generated by the stage when the stage is configured to automatically create the statements. The value cannot be edited but it can be copied to the clipboard.

User-defined
Edit custom SQL statements when the stage is configured to run user-defined statements. If the stage is first configured to automatically generate the statement and then switched to user-generated mode, the previously generated statement is copied to the User-defined tab so that you can use it as basis for further customization.

Before Configure the stage to run custom SQL statements after the stage has connected to the database and before it has processed any rows.

After Configure the stage to run custom SQL statements after the stage has processed all the rows and before the job ends.

Generated DDL
Displays the data definition language (DDL) statements that the stage has generated for dropping, creating or deleting table, depending on the table action specified for the stage.

User-defined DDL
Not supported. The stage only supports running auto-generated DDL statements.

Using job parameters for stage and link properties

You can set job parameters as environment variables or at the job level.

One option to consider for taking full advantage of the dynamic selection support is to define an environment variable for the database type at the project level and specify the default value that matches the database type currently in use. Then, add this environment variable as a job parameter to each job that uses the stage and specify the environment variable as the database type value in the stage. When at a later point a decision is made to utilize the same jobs to connect to a different database type, you would only need to change the default value of the environment variable at the project level and all the jobs will start using the new database type when they run the next time.

In addition to adding the environment variable as a job parameter, it is also possible to define job parameters directly at the job level. In either case it is possible to specify the default value to be used for the job. And when the job is started from InfoSphere DataStage Director, the value for the job parameter can be specified at that time.

When job parameters are used to specify the database type, then typically it can also be necessary to use job parameters for the database name, username, and password properties since these values will often not be the same for databases of different database types.

Defining a Dynamic RDBMS stage connection to the database

When a new Dynamic RDBMS stage is added to the server or parallel canvas, you must first open the **Stage** dialog box and specify the connection information for the database that the stage must connect to at runtime.

About this task

When the job starts, the stage connects to the specified database using the specified database credentials. As the job runs, the stage reads data rows from the database or writes data rows to it, depending on whether output or input links are defined for the stage, respectively.

Before indicating the particular database to which the stage should connect, you must specify the database type.

For the Dynamic RDBMS stage, set the properties on the **General** tab of the **Stage** page as described in the following task.

Procedure

1. Set the **Database Type** to your database type.
2. Set the **Connection Name** to the value that identifies the database that you want to connect to.
3. Set the **User ID** property to the value that identifies the user name for authentication and authorization with the database.
4. Set the **Password** property to the password value of the specified user name. The password is displayed and stored by the stage in encrypted form.

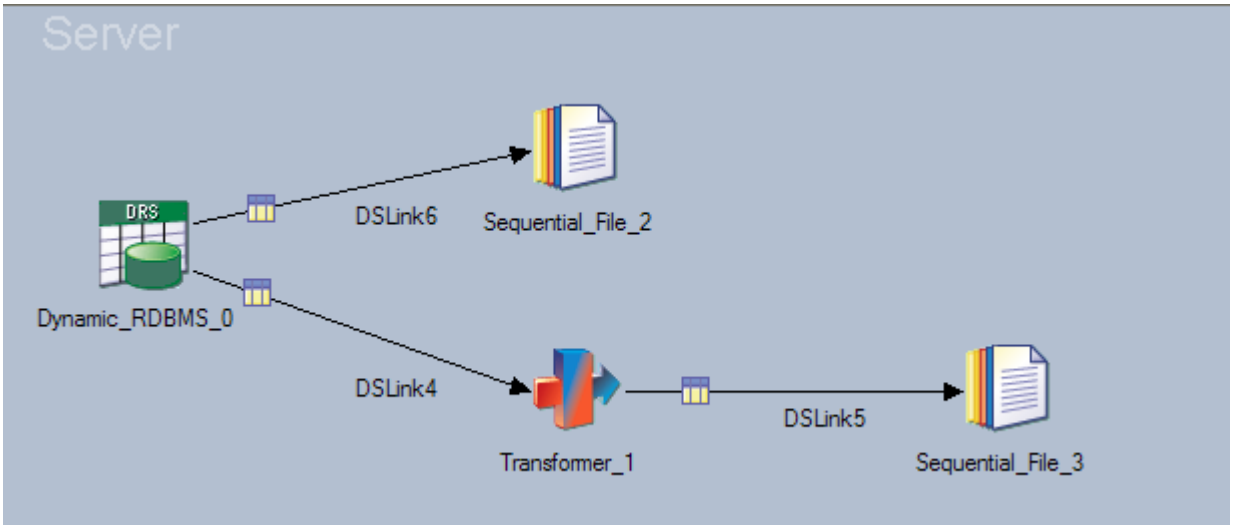
Reading data from the database

To configure the stage to read data from the database, define one or more output links on the stage and configure each link with the query to run on the database.

About this task

When the job runs, for each output link the stage runs the query specified on the link, fetches the result rows from the database, and delivers them to the downstream stage in the job.

The following figure shows the Dynamic RDBMS stage (**Dynamic_RDBMS_0**) in a server job configured to read data from a database and write that data to two files. To one of the files (**Sequential_File_2**) the data is written directly and to the other one (**Sequential_File_3**) the data is written after being transformed in the Transformer stage (**Transformer_1**).



The Dynamic RDBMS stage supports multiple output links on both the server and parallel canvases.

When multiple output links are defined for the stage all the links read data from the same database. The database connection definition applies to the stage as a whole and cannot be specified separately for individual links.

Auto-generated SELECT statements

You can configure the stage to automatically generate `SELECT` statements at run time.

For the Dynamic RDBMS stage to automatically generate the statement, you must set the **Query type** property to **Generated SQL query** on the **Output** > **General** page.

The following values need to be specified to generate the SELECT statement:

- Table name
- Column names on the output link
- WHERE clause (optional)
- Other clauses (optional)

On the Dynamic RDBMS stage the table name is specified in the **Table names (SQL FROM clause)** property on the **Output** > **General** page. It is possible to specify multiple table names to use in the auto-generated SELECT statement. The table names can be manually typed in as a comma-separated list of values or they can be specified by making a selection of imported table definitions from the metadata repository.

If only the table name values are specified, the schema under which the tables reside is the default schema for the user name defined for the stage. To specify a different schema, the table names need to be entered in fully-qualified form such as *schema_name*. *table_name*.

To treat the specified table names as case-sensitive values, they need to be enclosed in quoted identifiers of the target database. Typically double quotation marks are used a quoted identifiers.

The column names on the output link are used by the stage to produce the comma-separated list of columns in the generated SELECT statement. The names of the columns on the link are copied to the generated statement text. For each column you can specify the text to use in the auto-generated text instead of the column name. For example, this can be necessary when the column names in the table contain characters that cannot be displayed in the columns grid of the stage, to denote the table to which the column belongs when selecting from multiple tables or to preserve case sensitivity of the column names. The value is specified in the **Derivation** attribute for the column. This attribute is available for all column definitions on the **Columns** tab.

To preserve the case-sensitivity of the columns in the auto-generated statement in the Dynamic RDBMS stage, you must use the Derivation attribute for the columns and specify column names enclosed in quoted identifiers of the target table.

You can specify the WHERE clause to be appended to the auto-generated statement. This clause specifies the condition or filter to apply for producing the result data set. The WHERE clause is provided through **WHERE clause** property located on the **Selection** tab on the **Output** page of the stage dialog box. The WHERE keyword can be included in the specified value but this is not required.

You can also specify additional clauses (such as ORDER BY, GROUP BY and HAVING clauses) to be appended to the auto-generated statement. These clauses specify the sort order and grouping of rows in the result dataset. The additional clauses are provided through the **Other clauses** property located on the **Selection** tab under the **Output** page of the stage dialog box. The keywords that identify the clause types (such as ORDER BY, GROUP BY, and HAVING) must be included with the specified value.

The Dynamic RDBMS also generates the SELECT statement at runtime, but it also does this at design time and displays the generated statement in the stage dialog box. It is displayed on the **Generated** tab under the **SQL** tab on the **Output** page. The generated value cannot be edited but it can be selected and copied to the clipboard.

User-defined SELECT statements

You can manually specify a SELECT statement to run in the job.

The statement can be specified directly in the stage dialog box, or it can be specified in a file located on the InfoSphere DataStage Server engine machine and referenced by the stage.

In the Dynamic RDBMS stage you must set the **Query type** property on the **General** tab on the **Output** page to value User-defined SQL query. The statement text can then be specified on the **User-defined** tab under the **General** tab on the **Output** page.

The columns in the result set of the SELECT statement need to match the names of the columns on the output link for the stage. For example, if the SELECT statement retrieves the data from a database table with columns C1 and C2, the output link of the stage must also include columns C1 and C2.

When the statement text is stored in a file on the InfoSphere DataStage Server machine, the full path to the file can be specified in the stage and then the stage will open the file at runtime and reads the statement from the file.

To specify the file with SELECT statement in the Dynamic RDBMS stage you must set the **Query type** property on the **General** tab on the **Output** page to **User-defined SQL query file**. The file path can then be specified on the **User-defined** tab under the **General** tab on the **Output** page and must be in form FILE= *filepath* or {FILE} *filepath* where *filepath* is the fully-qualified path to the file on the InfoSphere DataStage Server engine machine. The Dynamic RDBMS stage does not support specifying a file path without FILE= or {FILE} prefix even if the **Query type** property is set to User-defined SQL query file.

Array read mode

When the stage runs the query at run time, it fetches all the rows in the result data set and delivers them on the output link.

You can specify how many rows to fetch at a time from the database. When more than one row is fetched at a time that operation is referred to as batch or array reads.

The array size is specified through the **Array size** property located on the **General** tab under the **Output** tab of the stage dialog box. The default value is 1, which effectively disables the batch or array read mode.

Writing data to the database

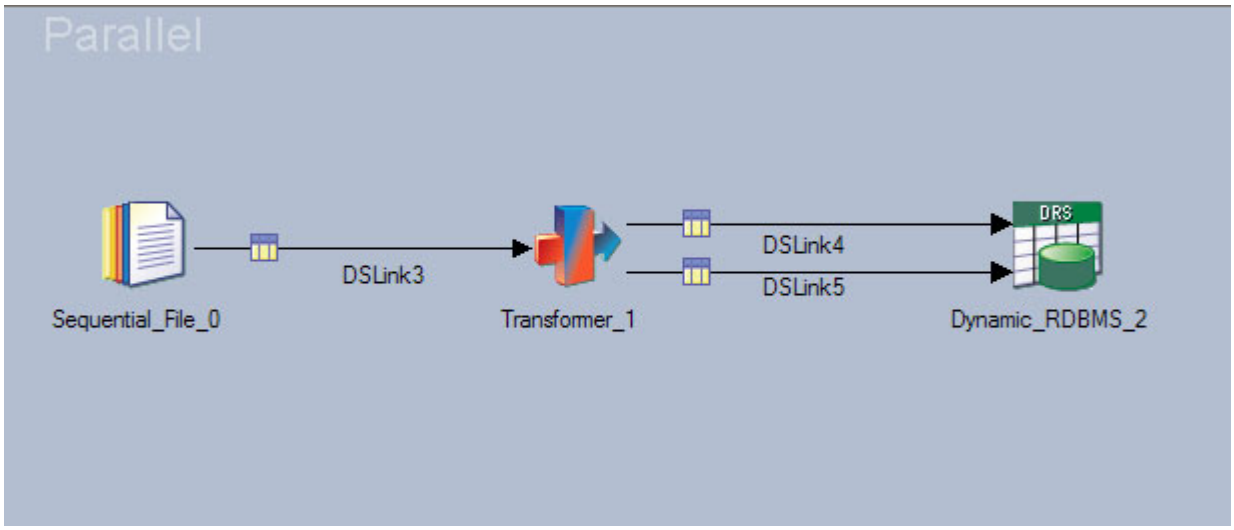
To configure the stage to write data to the database, you must define one or more input links on the stage and configure each link to perform the required write operation on the database.

About this task

When the job runs, for each input link the stage prepares the rows that it receives from the upstream stage in the job and writes them to the database.

When multiple input links are defined for the stage, they all must write data to the same database. The database connection definition applies to the stage as a whole and cannot be specified for individual links.

The following figure shows the Dynamic RDBMS stage (**Dynamic_RDBMS_2**) in a parallel job configured to write data to a database. The data records are read from a file (**Sequential_File_0**) and are passed to the Transformer stage (**Transformer_1**). The transformed data is then passed in two different formats on two separate links to the Dynamic RDBMS stage which can then write it to two different tables for example.



Write modes

The stage supports many different write modes that determine how the write operation is performed on the database.

The write mode is specified in the **Update action** property on the **General** tab of the **Input** page.

For certain write modes the stage needs to locate the matching rows in the target table for the input rows that it receives on the input link. A row in the target table is considered a match for the input row when it has the same values as the input row for the columns marked as Key column on the link. At least one column on the input link must be marked as a Key column for these write modes.

The following write modes are supported:

Insert rows without clearing
The rows are inserted in the table while preserving the existing content of the table.

Delete existing rows only
The stage deletes the rows in the target table that match the rows the stage receives on the input link.

Replace existing rows completely
The stage deletes all the rows in the target table that match the rows the stage received on the input link and inserts the rows from the link to the table.

Update existing rows only
The stage updates all the rows in the target table that match the rows the stage received on the input link. The values for the columns not marked as Key columns on the link are updated in the matched rows in the table with the value from the input row. The input rows for which no match is detected in the table are ignored.

Update existing rows or insert new ones
This mode is the same as the Update mode except the input rows for which no match is detected in the table are not ignored but are instead inserted to the table. The difference between this mode and the **Replace existing rows completely mode** is for the scenario when input row results in multiple matches in the table. This is possible since the columns marked as Key columns on the link do not necessarily need to correspond to an actual primary key or unique constraint in the table. When multiple matches are detected for an input row, the **Replace existing rows completely mode** mode will replace all the matching rows with that single input row while this mode will result in updating all the existing rows with values from the input row.

Insert new rows or update existing ones
For each input row the stage first tries to insert it into the database. If the insert operation fails due to unique constraint violation, the connector performs the update operation so all the rows in the table that match the input row are updated.

Insert new rows only
Is the same as the **Insert new rows or update existing ones** mode except the connector does not perform the followup update operation for the rows for which insert resulted in unique constraint violation.

Delete all rows
The entire content of the table is erased and all input rows are ignored. Therefore jobs that contain the stage configured with this mode typically ensure that no records or few records at most are actually delivered to the stage since they are ignored anyway.

Bulk insert
The stage takes advantage of the native bulk load support provided by the backend database to perform loading of rows into the table. When bulk load interface is supported by the database, it is typically faster to write data to the database using that interface than running standard `INSERT` SQL statements.

This write mode is supported only for database types that provide bulk load interface. It is not supported for ODBC database type.

Note: Some ODBC drivers can be configured to write data to the database in bulk load mode.

User-defined SQL
The stage runs the SQL statement or set of statements that is specified by the user.

User-defined SQL file
Is the same as **User-defined SQL** mode except the statements are not specified directly in the stage but in a file located on the InfoSphere DataStage Server machine.

When the stage automatically creates SQL statements for writing rows to the database, it references the specified table name and column names in the generated statements. Many databases treat the table and column names in the statements as uppercase values even if the actual values use mixed cases. To enforce the preservation of upper and lower cases in the specified table and column values, you can configure the stage to automatically enclose these values in quoted identifier characters of the database before inserting them in the generated statements. The quoted identifier character is typically the double-quote character.

To preserve case in the specified table and column names, you must select the **Enable case sensitive table/column name** property on the **General** tab of the **Input** page. By default this property is not selected.

User-defined SQL statements for writing data to the database

In some cases, the INSERT, UPDATE and DELETE statements that are generated automatically might not be suitable for the write operation that you want the connector to complete on the database. In these cases, you can enter the INSERT, UPDATE and DELETE statements manually.

In the Dynamic RDBMS stage the **Update action** property on the **General** tab of the **Input** page should be set to `User-defined SQL`.

In the Dynamic RDBMS stage the statements need to be specified on the **User-defined** tab under the **SQL** tab on the **Input** page.

The specified statements need to use bind parameters that the stage associates with the columns on the input link. The bind parameters are specified as question marks. The stage then internally translates the question marks to comply with the bind parameter syntax used by the backend database. When multiple statements are specified, the association between question marks and link columns is performed separately for each statement.

You can specify the statements in one of the following combinations:

A single INSERT statement
The stage associates question mark parameters in the statement with the columns on the link in the same order. The first question mark is associated with first column on the link, the second question mark with the second column on the link and so forth. In this case the stage operates in just like it was configured in **Insert** mode except the INSERT statement is specified manually.

A single UPDATE statement
The stage associates question marks in the `WHERE` clause of the statement with the columns on the link marked as Key columns. The first question mark in the `WHERE` clause is associated with the first key column on the link, the second question mark with the second key column and so forth. The question marks in the SET clause of the statement are associated with columns on the link that are not marked as Key columns. Again the first question mark in the SET clause is set with the first non-Key column on the link, the second question mark with the second non-Key column and

so forth. In this example, the stage operates similarly to **Update matching rows** mode except the UPDATE statement is specified manually.

A single DELETE statement
The stage associates question marks in the `WHERE` clause of the statement with the columns on the link marked as Key columns. The first question mark in the `WHERE` clause is associated with the first key column on the link, the second question mark with the second key column and so forth. The link columns that are not marked as Key columns are ignored. In this example, the stage operates just like it was configured in **Delete matching rows** mode except the DELETE statement is specified manually.

An INSERT statement followed by an UPDATE statement
The statements are separated by a semicolon character. In this case, the stage operates just like it was configured in **Insert then update** write mode except the statements are specified manually.

An UPDATE statement followed by an INSERT statement
The statements are separated by a semicolon character. In this case, the stage operates like it was configured in **Update then insert** write mode except the statements are specified manually.

A DELETE statement followed by an INSERT statement
The statements are separated by a semicolon character. In this case, the stage operates like it was set in **Delete then insert** write mode except the statements are specified manually.

Any combination of statements
The stage runs each statement separately.

However, if the Dynamic RDBMS stage is configured for the Oracle database type, this combination is not supported. The job fails with the error message indicating that only two statements can be specified.

In addition to specifying SQL statement directly in the stage, it is possible to store statements in a file on the InfoSphere DataStage Server machine and configure the stage to reference this file. The full path to the file is specified in the stage.

To configure the Dynamic RDBMS stage to read statements from a file the **Update action** property on the **General** tab of the **Input** page should be set to `User-defined SQL file`. The file path is then specified on the **User-defined** tab under the **SQL** tab on the **Input** page and must be in form `FILE=` *`filepath`* or `{FILE}` *`filepath`* where *filepath* is the fully-qualified path to the file on the InfoSphere DataStage Server machine.

Array write mode

The stage can be configured to send input rows to the database in array (batch) mode.

In this case the rows are not sent to the database one by one but are buffered by the stage in an array structure in memory and when the array is filled out (or there are no more rows on input) the whole array is sent to the database at once. You can specify the size of the array.

In the Dynamic RDBMS stage the write array size is specified in the **Array size** property on the **General** tab on the **Input** page. The default value is 1, which means that the array write mode is disabled by default for this stage.

Bulk load write mode

You can use the stage to load data to the target database by using native bulk load interfaces that are provided by the databases.

As the bulk load interfaces differ significantly among the databases, the properties used to configure the load operation also differ significantly.

When bulk load mode is selected for the stage, and a job parameter is used for the database type, the stage interface allows you to specify load properties for all database types for which bulk load is supported. When the job starts and you specify the actual database type to be used, the bulk load properties for that database type become effective and the bulk load properties for the remaining database types are ignored. That way you can take advantage of proprietary bulk load interfaces in different database types while keeping the option to delay the decision which database type to use until the job is actually started.

When the stage is configured to create data and control (configuration) files for the load operation, those files are created in the directory that is specified in the **Work directory** property. In the Dynamic RDBMS stage, this property is located under the **General** tab on the **Input** page.

Loading data to IBM DB2

These are the properties that can be set to customize the bulk load operation when the stage is configured for the IBM DB2 database type.

Load method

Specifies the mechanism to use for providing data to the database to be loaded to the target table. The following values are supported

Sequential file

The connector stores data to a file from which the loader loads it to the database

Named pipe

Default. The connector delivers the data directly to the loader.

Load immediate

Specifies whether to load the data as the job runs or to store data to a file to be loaded later. The allowed values are `Yes` and `No`. The default is Yes which means to load data during the job runtime.

Load from client

Specifies whether the load is done to a remote database or to the local database (installed on the InfoSphere DataStage Server machine).

Remove intermediate data file

Specifies whether to remove the data file after completing the load operation. This property is applicable only when the **Load method** property is set to **Sequential file**.

The supported values are `Yes` and `No`.

The default value is `Yes`, meaning the data file is removed after the data has been loaded.

File type of the data format

Specifies the data format to use in the data files when the **Load method** property is set to **Sequential file**.

The supported values are:

ASC ASCII data file with values specified in fixed size format.

DEL ASCII data file with delimiter used as the value separator.

This value is the default value.

IXF Proprietary data format called Integration Exchange Format (IXF), PC Version.

Cursor
A cursor declared against a Select of VALUES statement.

LOB path
Specifies the path to the directory with data files that contain LOB values that need to be loaded to the target table. There is no default value.

File type modifier
Specifies the modifier options (as free-form text) to be provided for the MODIFY load parameter.

The default value is `lobsinfile noheader`.

Method
Specifies how to perform the resolution of columns to which the data should be loaded based on the input data contents.

The supported values are:

Default
The stage automatically determines the method to use based on the specified data format.

This method is supported for `DEL` data format only.

Locations
The stage uses the start and end columns numbers specified by the user.

Names
The stage uses column names specified by the user

Positional
The stage uses the fields numbers of the input data fields specified by the user

Column-start Column-end
Applies when the **Method** property is set to `Locations`. It specifies a comma-separated list of *start end* pairs of column numbers for the columns in the input data file from which to load data to the target table. The value is in format: n1 n2, n3 n4, etc. The specified numbers represent byte offsets for the respective columns from the beginning of each data row. There is no default value.

Column name
Applies when the **Method** property is set to `Names`. It specifies a comma-separated list of column names in the input data file from which to load data to the target table. There is no default value.

Column position
Applies when the **Method** property is set to `Positional`. It specifies a comma-separated list of field numbers (starting with 1) of the input data fields to be loaded. There is no default value.

Null indicators
Applies when the **Method** property is set to `Locations`. It specifies a

comma-separated list of column numbers for the null indicator fields. Column numbers represent byte offsets of the null indicator fields from the beginning of each data row. There is no default value.

Insert-column
Specifies a comma-separated list of column names in the target table to which to insert data. Double quotation marks should be used around column names that contain spaces. There is no default value.

Datalink specification
Provides column specifications for DATALINK columns. There can be one column specification for each DATALINK column. Column specifications are enclosed by parentheses. Each column specification consists of one or more of DL_LINKTYPE, prefix and DL_URL_SUFFIX specifications. The prefix specification can be either DL_URL_REPLACE_PREFIX or DL_URL_DEFAULT_PREFIX specification. There is no default value. This property is deprecated as the DATALINK SPECIFICATION option is not supported for LOAD utility in DB2 9.1 and later.

Without prompting
Specifies whether the list of specified data files contains all the files that need to be loaded and that the devices or directories listed are sufficient for the entire load operation. The supported values are `Yes` and `No`.

The default value is `No`.

Rows buffer size
Specifies the size of the buffer in kilobytes to use for the rows from the input link.

Load mode
specifies the mode in which to load the data.

The supported values are:

Insert Appends new data to the table without changing the existing data in the table.

Replace
Deletes all the data from the table before loading the new data.

This value is the default value.

Restart
Restarts the load operation following the interruption of the previous load operation.

Terminate
Terminates the load operation that was previously interrupted.

Save count
Specifies the number of rows to load before establishing consistency point. The default value 0 is used to specify that no consistency points are established, unless determined to be necessary by the load utility.

Row count
Specifies the total number of initial rows to load. The default value 0 specifies that all the rows in the input data file should be loaded.

Restart count
Specifies the number of rows to skip before starting to load rows. This property should be used when the previous load attempt failed with some records committed to the target table.

The default value 0 specifies that not records should be skipped and to start loading from the first row.

Restart phase
Specifies the phase at which to restart the load operation. Build and Delete values should not be specified if Insert or Replace values are specified for the Load mode property.

The supported values are:

Load Start the operation in the Load phase.

Build Start the operation in the Build phase.

Delete Start the operation in Delete phase.

Warning count
Specifies the number of warnings after which to stop the load operation. The default value is 0 which means the load operation proceeds regardless of the number of warnings that were reported.

Indexing mode
Specifies whether to rebuild indexes or to extend them incrementally. The supported values are:

AUTOSELECT
The load utility automatically chooses the most optimal option.

This value is the default value.

REBUILD
The indexes are rebuilt after the load

INCREMENTAL
The indexes are maintained (incrementally built) during the load

DEFERRED
The decision is deferred

Load message file name
Specifies the path to the local file to use for storing warning and error messages during the load operation.

There is no default value.

Directory for temporary files
Specifies the path to the directory that DB2 uses for storing temporary files that it creates for the load operation.

Exception table name
Specifies the name of the table to which to insert rows in error.

There is no default value.

Statistics
The types of statistics to collect for the table.

The collection of statistics is not supported if the **Load mode** property is set to **Insert** or **Replace**.

The supported values are:

TableStats
Table statistics are gathered.

TableAndIndexStats
Table and index statistics are gathered.

Index statistics
Index statistics are gathered.

TableAndDistributedStats
Table and distributed statistics are gathered.

TableAndDistributedStatsAndBasicIndexes
Table, distributed and basic index statistics are gathered.

ExtendedStatsForIndexOnly
Extended index statistics are gathered.

ExtendedStatsForIndexesAndBasicTableStats
Extended index statistics and basic table statistics are gathered.

AllStatistics
All statistics are gathered.

NoStatistics
None of the statistics are gathered.

This value is the default value.

Non Recoverable
Specifies whether the transaction for the data load operation unrecoverable. The supported values are `Yes` and `No`. The default value Yes indicates that the data load operation is unrecoverable, meaning that the loaded data cannot be recovered by a subsequent roll forward operation.

Data buffer size
Specifies the number of 4-kilobyte pages in the buffer to use for transferring data within the load utility. The default value 0 specifies that the load utility should automatically determine the optimal value.

Sort buffer size
Specifies the number of 4-kilobyte pages in the buffer to use for sorting index keys in the load utility when the **Indexing mode** property is not set to value `Deferred`. The default value 0 specifies that the load utility should automatically determine the optimal value.

Working directory
Specifies the optional working directory for sorting index keys. There is no default value which indicates that sqllib/tmp directory should be used. This property is deprecated as DB2 changed the sort operations to spill into a bufferpool associated with the temporary table space.

CPU parallelism
Specifies the number of processes or threads that the load utility will create for parsing, converting, and formatting records when building table objects. The default value 0 specifies that the load utility should automatically choose the optimal CPU parallelism value based on the current environment.

Disk parallelism
Specifies the number of processes or threads that the load utility will create for writing data to the table space containers. The default value 0 specifies that the load utility should automatically choose the optimal disk parallelism value based on the current environment.

Tracing level
Specifies the level of tracing for the messages added to the log. The actual value specified for the property is a combination of the supported values obtained by adding them together to produce the effective tracing level.

The supported values are:

0 No tracing.

1 Tracing of important events.

2 Tracing performance messages.

4 Tracing function messages.

Copy loaded data
Specifies whether to save a copy of the loaded data.

The supported values are:

No No copy is made. This is the default value.

Yes C copy is made by the user-specified library.

Use TSM
A copy is made using Tivoli® Storage Manager.

Use ADSM
A copy is made using ADSTAR Distributed Transaction Manager.

This option is deprecated as it is not supported starting with DB2 9.1.

Copy To device/directory name
Specifies the name of the device or directory to which to save a copy of the loaded data. There is no default value. This property is applicable when the **Copy loaded data** property is not set to value `No`.

Copy Load library name
Specifies the name of the shared library containing the backup and restore I/O functions to be used for saving a copy of the loaded data. There is no default value. This property is applicable when the **Copy loaded data** property is set to value **Yes**

Allow access mode
The access level to allow to other applications for the target table to which the data is loaded.

The supported values are:

No The LOAD utility locks table for exclusive access during the load.

Read The LOAD utility locks the table in shared access mode so that other applications can read data from it while the load is taking place.

Use table space to allow read access
Specifies the optional tablespace to use if rebuilding indexes. A shadow copy of the index is built in the specified tablespace and then copied to the original tablespace at the end of the load.

There is no default value.

Check pending cascade
Specifies whether to automatically set Integrity pending state to all the descendent tables.

The supported values are:

Deferred
Only the table that is loaded is placed in the Integrity pending state.

This value is the default value.

Immediate
The Integrity pending state for foreign key constraints is immediately extended to all descended foreign key tables.

Lock with force
Specifies whether the load utility should be allowed to force off other applications that hold conflicting locks on the target table in order to acquire the necessary locks for the load operation. The supported values are `Yes` and `No`. The default value is `Yes`.

Partitioned DB configuration
Specifies whether the data is loaded to a table distributed across multiple database partitions. The supported values are `Yes` and `No`.

The default value is `No`.

HOSTNAME
Specifies the host name of the machine where the data file resides. If the value is not provided, the value "nohost" is used.

This property is deprecated as this option has been deprecated starting with DB2 9.1.

FILE_TRANSFER_CMD
Specifies the name of the executable to use to provide data to the load utility.

There is no default value.

This property is deprecated as this option has been deprecated starting with DB2 9.1

PART_FILE_LOCATION
Specifies the path of the directory with partitioned files.

There is no default value.

OUTPUT_DBPARTNUMS
Specifies the comma-separated list of database partitions to which to load the data. The list must be enclosed by parenthesis. The word "to" can be used to specify a range or partitions instead of a single partition, for example: (0, 2 to 10, 15).

There is no default value.

PARTITIONING_DBPARTNUMS
Specifies the comma-separated list of database partitions to be used in the distribution process. The list must be enclosed by parenthesis. The word "to" can be used to specify a range or partitions instead of a single partition number, for example: (0, 2 to 10, 15).

There is no default value.

MODE
Specifies the mode to use for loading data to the table in a partitioned database.

The supported values are:

PARTITION_AND_LOAD
Partition and load the data.

PARTITION_ONLY
Partition the data only.

LOAD_ONLY
Load the data.

This value is the default value.

LOAD_ONLY_VERIFY_PART
Load the data for verification only.

ANALYZE
Analyze the data only.

MAX_NUM_PART_AGENTS
Specifies the maximum number of partitioning agents in a load session.

The default value is `25`.

MAX_NUM_PART_AGENTS
Specifies the mode for handling errors that occur on individual database partitions.

The supported values are:

SETUP_ERRS_ONLY
Handle setup errors only.

LOAD_ERRS_ONLY
Handle load errors only.

This value is the default value.

SETUP_AND_LOAD_ERRS
Handle setup and load errors.

NO_ISOLATION
Handle no isolation level errors.

STATUS_INTERVAL
Specifies the amount of data to load in megabytes before issuing a progress message.

The valid values are in the range of **1** to **4000**.

The default value is `100`.

PORT_RANGE
Specifies the range of TCP ports used to create sockets for internal communications. The format is: (lower-port,higher-port).

The default value is `6000,6063`.

CHECK_TRUNCATION
Specifies whether to check for data truncations on input/output.

The valid values are `Yes` and `No`.

The default value is `No`.

MAP_FILE_INPUT
Specifies the name of the input distribution map file name to use when the **MODE** property is set to `ANALYZE`.

There is no default value.

MAP_FILE_OUTPUT
Specifies the name of the output distribution map file name to use when the **MODE** property is set to ANALYZE.

There is no default value.

TRACE
Specifies the number of records to trace when a review of a dump of the data conversion process and the output of the hashing values is required.

The default value is 0.

NEWLINE
Specifies whether to force checking for newline character at the end of each record.

The option is valid when non-delimited ASCII format file is loaded and the **[ASC]** property is set and the **reclen** file type modifier is specified.

The supported values are Yes and No.

The default value is No.

DISTFILE
Specifies the name of the database partition distribution file that the LOAD utility should generate. If the value is not specified, the file is not generated.

There is no default value.

OMIT_HEADER
Specifies whether to prevent inclusion of the distribution map header in the distribution file.

The supported values are Yes and No.

The default value is No.

RUN_STAT_DBPARTNUM
Specifies the number of the partition for which to collect statistics if the stage was configured to collect statistics. The value -1 specifies to collect statistics on the first database partition in the output database partition list.

The default value is -1.

Loading data to Oracle

These are the properties that can be set to customize the bulk load operation when the stage is configured for the Oracle database type.

Schema name
Specifies the name of the schema (owner) in which the target table resides.

There is no default value.

If no value is specified the table is assumed to reside in the schema of the currently connected user.

Partition name
Specifies the name of the table partition to which to load data.

There is no default value.

If the value is not specified, the data is loaded to the entire table.

Max Record Number
Specifies the maximum number of input records in a batch.

The default value is 100.

Load mode
Specifies the method to use to load the data in the target table.

The supported values are:

Automatic
Loads the data directly to the target database table using Direct Path Oracle interface.

Manual
Creates Oracle SQL*Loader control file and data file which can later be used to perform the load with Oracle SQL*Loader utility.

Control file name
Specifies the name of the Oracle SQL*Loader control file that the stage creates when the **Load mode** property is set to value Manual.

There is no default value.

Data file name
Specifies the name of the Oracle SQL*Loader data file that the stage creates when the **Load mode** property is set to Manual.

There is no default value.

Delimiter
Specifies the character used to delimit field values in the input data.

The default values is comma character (,).

Preserve Blanks
Specifies whether to preserve trailing blanks in the input text values or to truncate them.

The supported values are:

Yes Preserve the trailing blanks.

This value is the default value.

No Truncate trailing blanks.

Case sensitive column names
Specifies whether the input link column names should be treated as case sensitive values.

The supported values are:

Yes The column names should be treated as case-sensitive values.

No All characters in the input link column names should be treated as uppercase characters.

This value is the default value.

In the manual load mode the stage does not write data to the database table directly, but instead creates Oracle SQL*Loader control and data files to be used with Oracle SQL*Loader to load the data to the table. The format of these two files is determined internally by the stage and can change in future releases of the product. For this reason it is not recommended to design jobs that rely on the current format of these files.

If the control file name is not specified, the stage will create the control file name consisting of the specified Oracle service name, followed by the underscore character, followed by the specified name of the table to which the data is loaded, followed by the `.ctl` extension.

If the data file name is not specified the stage will use the same logic to produce the data file name except instead of the `.ctl` extension, the `.dat` extension will be used for the file name.

Loading data to Informix

These are the properties that can be set to customize the bulk load operation when the stage is configured for Informix database type.

Truncate table then load
Specifies whether the table should be cleared before loading the data.

The supported values are:

Yes Do not clear the table before loading the data,

No Do not clear the table before loading the data,

This is the default value.

Rows to flush
Specifies the number of rows to load before issuing a flush.

The default value is `1,000`.

Insert buffer size
Specifies the size of the insert cursor buffer.

The *FET_BUF_SIZE* environment variable determines the size of the buffer for all select, fetch, and insert cursors in all Informix clients. Each client can override the *FET_BUF_SIZE* value.

The allowed values are between `4096` and `32,767`.

Values outside of this range are ignored, except for the default value `0` which is used to represent system default.

Lock mode
The lock mode to use for the target table.

The supported values are:

Shared
The table is locked in shared mode so other applications can read the data from the table while it is being loaded.

Exclusive
The table is locked in exclusive mode so other applications cannot access the table while it is being loaded.

None No lock is obtained on the table.

This is the default value.

Before-link routine
Specifies an external routine to execute before processing the data on the link. The value represents the command to execute by the operating system. The command can be a program executable, a UNIX shell script, or a DOS batch file. It must be in a format understood by the operating system. The stage does not parse or perform syntax checking on the specified value.

After-link routine
Same as Before-link routine except this it is executed after processing the data on the link.

Default datetime format
Specifies the default datetime format to use when creating Timestamp columns and while inserting Timestamp field values into Timestamp columns.

Valid datetime formats mirror Informix DDL syntax for a datetime type, for example `Hour to Fraction(2)`. The datetime type specified for this property should be the same as or a subset of the Datetime type defined for the actual column in the table. Any format specified in the Description attribute of the column on the link overrides the value in this property. The widest format is `Year to fraction(5)`.

The default value is `Year to fraction(3)`.

Loading data to Microsoft SQL Server

These are the properties that can be set to customize the bulk load operation when the stage is configured for the Microsoft SQL Server database type.

Loading data to Microsoft SQL Server is supported only if InfoSphere DataStage engine is installed on a Microsoft Windows machine.

For Linux or UNIX operating systems, you can still select Bulk insert write mode in the Dynamic RDBMS stage configured for Microsoft SQL Server database, but the job with that stage will fail to run successfully and an error message will be reported in the job log stating that the stage supports bulk load operation to Microsoft SQL Server databases only on Windows.

The following is the list of properties that can be set to customize the bulk load operation in the Dynamic RDBMS stages configured for Microsoft SQL Server database type.

Commit size
Specifies the number of rows to load before committing them to the target table. The rows are loaded in delayed mode, and are stored to the table only after commit call is issued on the database load interface. The default value is **100** which means that the rows are committed to the target table in batches of 100 rows.

Use source identity data
Specifies whether values for identity column are provided with the data or if the database dynamically created them.

The supported values are:

Yes The identity column values are supplied with the data.

No The identity column values are generated by the database.

This is the default value.

Keep nulls
Specifies whether to preserve NULL values during the load.

Valid values are:

Yes NULL values are inserted in the column.

No NULL values in the data are replaced with the default value for the column.

This is the default value.

Check constraints
Specified the method for handling table constraints.

The supported values are:

Yes The constraints are checked during the data load and the rows that violate the constraints are not loaded.

No The constraints are ignored.

This is the default value.

Bind names
Specified the mechanism to use for associating columns on the link with the columns in the target table.

The supported values are:

Yes The association is done by name. Columns on the link are matched with columns in the target table by name, irrespective of the column positions.

This is the default value.

No The association is done by position. The first column on the link is associated with the first column in the table, the second column on the link with the second column in the table, and so forth.

Load action
Specifies the action to perform on the table before loading the data to the table.

The supported values are:

Clear table then load
Deletes rows from the table.

Truncate table then load
Issues truncate operation on the table.

This is the default value.

Append to the table
Preserves the existing content of the table.

Transaction isolation level
Specifies the transaction isolation level.

The supported values are:

ReadUncommitted
Dirty reads, non-repeatable reads and phantom reads are possible.

ReadCommitted
Non-repeatable and phantom reads are possible.

This is the default value.

RepeatableRead
Phantom reads are possible.

Serializable
The highest isolation level, none of the dirty reads, non-repeatable reads or phantom reads are possible.

Tracing level
Specifies the level of tracing information added to the log.

The actual value specified for the property is the sum of the flag values from the list. For example the value 3 means that property values and performance indicators are logged.

The supported values are:

0 No tracing.

This is the default value.

1 Traces stage property values.

2 Traces performance indicators.

4 Traces important events.

Before load statement
Specifies a semicolon-separated list of SQL statements to execute after connecting to the database and before loading any data to the target table.

There is no default value.

Continue before load
Specifies the action to take if a statement in the **Before load statement** set of statements fails.

The supported values are:

Yes Logs any error as a warning and the next statement is processed. Each successfully completed statement is committed.

This is the default value.

No If any statement fails to execute successfully, the transaction is rolled back and the job fails. A single commit is issued at the end if all the statements complete successfully.

After load statement
Specifies a semicolon-separated list of SQL statements to execute after all the data has been loaded and before the job ends.

There is no default value.

Continue after load
Same as Continue before load except it applies to the statements in the **After load statement** set of statements.

Loading data to Sybase

These are the properties that can be set to customize the bulk load operation when the stage is configured for Sybase databases.

Before load stored procedure
Specifies the name of the stored procedure in the database to run after connecting to the database and before loading any data to the target table.

There is no default value.

After load stored procedure
Specifies the name of the stored procedure in the database to run after loading all the data to the target table and before the job ends.

There is no default value.

Batch size
Specifies the number of rows to load before issuing a commit call. The rows are loaded in delayed mode and are committed to the target table when the stage issues commit call on the database load interface.

The default value is **0** which means that all the loaded rows are committed at once. This property corresponds to Sybase BCP **-b** switch.

Packet size
Specifies the size of the data packet in bytes to send to the server. The supported values are in the range from **512** to **65,535**. The default value is **4,096**.

Use source identity data
Specifies whether to override the automatically generated Identity column. This property is the equivalent of the Sybase BCP **-E** switch.

The supported values are:

Yes Overrides the identity column with values provided in the data.

This is the default value.

No Uses the database generated values.

Database lookup operation

You can configure the stage to complete a lookup operation on the database for each input record (referred to as the key record) and return rows that match the criteria that are specified by that record.

The lookup operation is performed by running a parameterized SELECT statement which contains a `WHERE` clause with parameters that are associated with the columns marked as Key columns in the records that represent key records for the lookup.

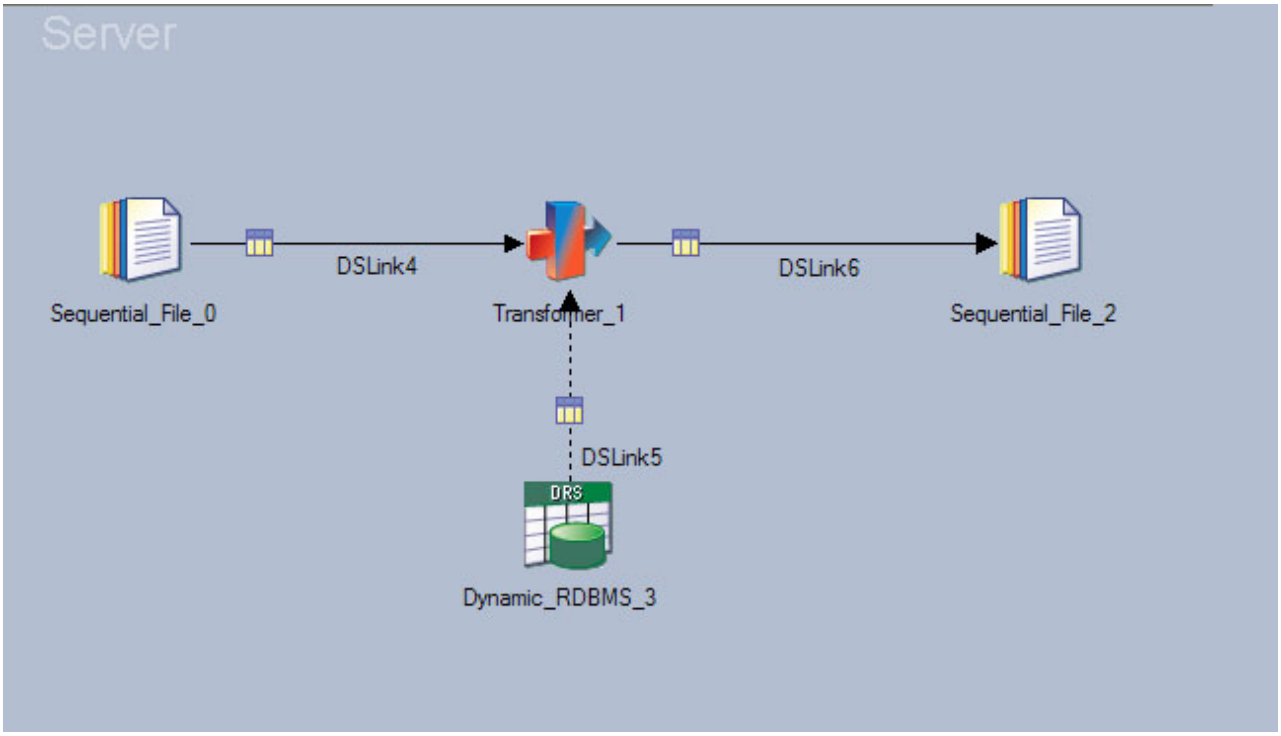
On server canvas the job configuration for the database lookup requires that the Transformer stage is used in combination with the database stage. The Transformer stage has an input link on which the key records arrive to be used as input for the lookup query. It also has one or more reference links coming from the database stage. The database stage is provided with input key records on this link. For each input key record, the database stage runs the parameterized `SELECT` statement with the key record values used in the `WHERE` clause and provides the corresponding matching records to the Transformer stage. Those records are then processed and routed by the Transformer stage to one or more of its output links to be further processed by the downstream stages in the job.

In some cases the `SELECT` lookup statement may return multiple record matches. The user can specify whether the stage should log a message when this happens.

In the Dynamic RDBMS stage this is specified through the Disable warning message for reference lookup property under General tab on the Output page. By default this property is checked meaning that the warning messages are not logged. When the property is unchecked, the warning messages are written to the job log.

The following figure shows a Dynamic RDBMS stage that is configured for lookup operation in a server job. The records from the input file (**Sequential_File_0**) are passed through the Transformer stage (**Transformer_1**) to the Dynamic RDBMS

stage (**Dynamic_RDBMS_3**) which uses their key column values as input for performing lookup operation on the database. The results of the lookup operation are written to the output file (**Sequential_File_2**).



On the parallel canvas the lookup operation is not supported by the Dynamic RDBMS stage.

The configuration for the database lookup on the parallel canvas is similar to one on the server canvas. The Lookup stage is used instead of the Transformer stage.

Creating, dropping, and deleting database tables at runtime

You can configure the stage to create, replace or truncate the tables in the database before writing data rows to it.

The following table actions are supported:

Append
The table referenced by the stage must exist in the database.

This is the default table action.

In the Dynamic RDBMS stage this table action is specified by setting the **Create table action** property under **General** tab on the **Input** page to `Do not create target table`.

Create The stage creates the table in the database.

The `CREATE TABLE` statement is generated automatically by the stage from the specified table name and column definitions on the input link.

In the Dynamic RDBMS stage, this table action is specified by setting the **Create table action** property under **General** tab on the **Input** page to `Do not drop target table`.

Replace
The stage drops the table (if it exists) from the database and creates the table again.

The `CREATE TABLE` and `DROP TABLE` statements are generated automatically by the stage from the specified table name and column definitions on the input link.

In the Dynamic RDBMS stage, this table action is specified by setting the **Create table action** and **Drop table** properties under **General** tab on the **Input** page to `Generate DDL`.

Delete The stage deletes all the rows from the table before writing new rows to the table.

In the Dynamic RDBMS stage, this table action is specified by setting the **Update action** property under **General** tab on the **Input** page to either of the following values:

Clear table then insert rows
The stage generates and runs the `DELETE TABLE` statement.

Truncate table then insert rows
The stage generates and runs the `TRUNCATE TABLE` statement, when this is supported for the selected database type.

Generating SQL statements in the connector at design time

You can configure the connector to generate SQL statements at design time in their statement properties.

Before you begin

Create a job that includes a connector as a source or target.

About this task

You can generate the SQL statement text only for those statement properties that have the **`Generate SQL statement`** option in the Build list.

Note: Under some circumstances, the connector requires a connection to generate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output or input link, depending on the type of job that you create.
3. Set **`Generate SQL at runtime`** to `No`.
4. In the **`Table name`** property, type the name of the table for the SQL statement.
5. For jobs in target context (input links), select the type of statement you want to generate in the **`Write mode`** property.
6. On the **Columns** page, define the columns to use in the SQL statement.
7. Click the **Properties** tab.
8. Click the **Build** button that is associated with the statement property, and select **`Generate SQL statement`** from the list.

Note: The **Generate SQL statement** option will only be available for statements which that connector supports generating at design time. In some cases a connector may only support generating the SQL at runtime during job execution.

9. Click **OK** to save the job.

Validating SQL statements in the connector at design time

After you generate or write a SQL statement, you can validate the statement during job design.

About this task

You can validate the SQL statement text only for those statement properties that have the **Validate SQL** option in the Build list.

Note: Under some circumstances, the connector requires a connection to validate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

Procedure

1. Save the job.
2. Click the **Build** button that is associated with the statement property, and select **Validate SQL**. The **Validate SQL** option is enabled only if the statement property contains a value and this option will only be available for statements which the target RDBMS supports validating.

Results

The connector validates the SQL statement by preparing the statement with the RDBMS it supports. If the SQL contains error, an error message is shown.

Data type support

Different databases support proprietary data types, which the stage needs to convert to and from the InfoSphere DataStage data types that are specified for the column on the stage links.

For example, when the stage is configured to create a table in the target database, it constructs the `CREATE TABLE` statement and for each column on the link it specifies the data type to be used in the target table. For some InfoSphere DataStage data types this mapping is supported and for some it is not.

When the job runs and the stage reads data from the database and writes data to the database, it needs to convert data between the column data types in the database and the InfoSphere DataStage data types for the corresponding columns on the link. The following topics provide mapping details for each of the supported database types.

The topics include conversion tables that show how the stages map InfoSphere DataStage data types to target database types when constructing the `CREATE TABLE` statement.

The tables should be used as a guide when choosing which InfoSphere DataStage data type to use for columns on the link depending on the column definitions in the database table. For a particular database type check if that data type appears in the last column in the conversion table. If the database type is not listed in the conversion table, choose the InfoSphere DataStage type that is the closest possible match for that database data type. Ensure that the *Length*, *Scale*, *Extended*, and *Nullable* attributes for the column on the link are set to values that match the column definition in the database.

When starting with an empty link and when the selection of column types is not dictated by the other stage on the link, the preferred option for specifying columns on the link is to use the metadata import wizards available in InfoSphere DataStage and QualityStage Designer. The wizards allow you to import target table definition into the metadata repository. Once imported, the table definition can be applied to the links of the stage. This can be done by clicking the Load button on the Columns tab in the stage dialog, or the table definition can be dragged and dropped from the Repository view directly onto the link.

For the Dynamic RDBMS stage the preferred wizard is the Plug-in Meta Data Definitions wizard.

For the link columns of `Decimal` and `Numeric` data types it is always necessary to specify the Length attribute. This attribute represents the decimal precision. The *Scale* attribute represents the decimal scale and is optional. When the *Scale* attribute is not specified then the scale of zero is assumed.

It is required to specify the *Length* attribute for the Dynamic RDBMS stage for the link columns of the following data types:

- `Char`
- `VarChar`
- `LongVarChar`
- `NChar`
- `NVarChar`
- `LongNVarChar`
- `Binary`
- `VarBinary`
- `LongVarBinary`

IBM DB2 data type support

When a job runs, the stage maps InfoSphere DataStage data types to IBM DB2 data types.

Table 1. Mapping of InfoSphere DataStage data types to IBM DB2 data types

SQL type	Length	Scale	Extended	IBM DB2 Column Definition
BigInt	n/a	n/a	any	BIGINT
Binary	*n*	n/a	n/a	CHAR(*n*) FOR BIT DATA[1]
Bit	n/a	n/a	n/a	not supported
Char	*n*	n/a	blank	CHAR(*n*)[1]
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATE
Decimal	*p*	*s*	n/a	DECIMAL(*p,s)*

Table 1. Mapping of InfoSphere DataStage data types to IBM DB2 data types (continued)

SQL type	Length	Scale	Extended	IBM DB2 Column Definition
Double	*p*	n/a	n/a	FLOAT(*p*)[1]
Float	*p*	n/a	n/a	FLOAT(*p*)[1]
Integer	n/a	n/a	any	INTEGER
LongNVarChar	*n*	n/a	n/a	LONG VARCHAR[2]
LongVarBinary	*n*	n/a	n/a	LONG VARCHAR FOR BIT DATA
LongVarChar	*n*	n/a	blank	LONG VARCHAR[3]
LongVarChar	*n*	n/a	Unicode	LONG VARCHAR[2]
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	VARGRAPHIC(*n*)[1]
Numeric	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	*s*	any	TIME
Timestamp	n/a	*s*	any	TIMESTAMP
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	VARCHAR(n) FOR BIT DATA[1]
VarChar	*n*	n/a	blank	VARCHAR(*n*)[1]
VarChar	*n*	n/a	Unicode	(VARCHAR*n*)

Table notes:

1. If the Length is left blank the Dynamic RDBMS stage assumes the length of 15. Depending on the specified Length p, IBM DB2 maps the FLOAT(*p*) data type to REAL or DOUBLE native data type.
2. If the Length n is greater than 32700, the Dynamic RDBMS stage creates DBCLOB(*n*).
3. If the Length n is greater than 32700, the Dynamic RDBMS stage creates CLOB(n).

ODBC data type support

When a job runs, the stage maps InfoSphere DataStage data types to target database types.

Multiple data types can be listed in the table because the actual target data type depends on the ODBC driver in question. In some instances, the ODBC driver will not accept the target data type specified by the stage when creating the table. For example, the Dynamic RDBMS stage will map the*Binary* source data type to the *CHAR(n) FOR BIT DATA* target data type, which will be accepted by DB2 driver but not by Oracle.

The database can accept the data type specified by the stage, but will then internally convert it to a more suitable native data type.

For example, the Dynamic RDBMS stage will map *Integer* source data type to *INTEGER* target data type, which Oracle will accept but will then convert it to *NUMBER(38)* native data type.

The generic data type is the type that the stage specifies in the auto-generated `CREATE TABLE` statement. The mapping from this data type to the actual database

type in the backend database is performed by the ODBC driver. If the mapping is not supported, the `CREATE TABLE` statement fails.

Table 2. Mapping of InfoSphere DataStage data types to ODBC data types

SQL type	Length	Scale	Extended	Generic Column Definition
BigInt	n/a	n/a	any	BIGINT
Binary	*n*	n/a	n/a	CHAR(*n*) FOR BIT DATA
Bit	n/a	n/a	n/a	Not supported
Char	*n*	n/a	blank	CHAR(*n*)
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATE, DATETIME
Decimal	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Double	*p*	n/a	n/a	FLOAT(*p*)
Float	*p*	n/a	n/a	FLOAT(*p*)
Integer	n/a	n/a	n/a	INTEGER
LongNVarChar	*n*	n/a	n/a	VARCHAR(n), VARCHAR2(n) NTEXT, LONG
LongVarBinary	*n*	n/a	n/a	IMAGE, LONG
LongVarChar	*n*	n/a	blank	TEXT, LONG
LongVarChar	*n*	n/a	Unicode	TEXT, LONG
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	NVARCHAR(*n*)
Numeric	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	any	any	TIME, DATETIME, DATE
Timestamp	n/a	any	any	TIMESTAMP, DATETIME, DATE
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	VARCHAR(*n*) FOR BIT DATA
VarChar	*n*	n/a	blank	VARCHAR(*n*)
VarChar	*n*	n/a	Unicode	VARCHAR(*n*)

Oracle data type support

When a job runs, the stage maps InfoSphere DataStage data types to Oracle data types.

The Oracle DATE data type supports year, month, day, hour, minute and second portions. On the other hand InfoSphere DataStage Date data type supports only year, month and day portions. Oracle *TIMESTAMP* and InfoSphere DataStage *Timestamp* data types support year, month, day, hour, minute, second, and fractional seconds.

Oracle does not allow empty string values for columns of character Oracle data types of variable size (such as *VARCHAR2*, *NVARCHAR2*, *CLOB*, and *NCLOB*). InfoSphere DataStage supports empty string values. When an empty string value is provided to Oracle, it writes it to the database as a `NULL` value. This is important to

note when considering the nullable attribute of the columns on the link. For example, an empty string value is valid for a link column for which Nullable attribute is set to No. However, it is invalid for an Oracle column that does not allow NULL values (marked as NOT NULL).

Similar consideration needs to be made when writing values to Oracle columns of character data types of fixed size (such as *CHAR* and *NCHAR)*. When an empty string value is provided to Oracle to write to these columns, Oracle will not pad the value with spaces; Oracle will write NULL value to the column. If the table column does not allow NULL values that will result in error.

Table 3. Mapping of InfoSphere DataStage data types to Oracle data types

SQL type	Length	Scale	Extended	Oracle Column Definition
BigInt	n/a	n/a	any	NUMBER(38)
Binary	n	n/a	n/a	RAW(n)[1]
Bit	n/a	n/a	n/a	Not supported
Char	*n*	n/a	blank	CHAR(n)
Char	*n*	n/a	Unicode	CHAR(n)
Date[1]	n/a	n/a	n/a	DATE
Decimal	*p*	*s*	n/a	NUMBER(*p*,*s*)
Double	*p*	n/a	n/a	FLOAT(*p*)
Float	*p*	n/a	n/a	FLOAT(*p*)
Integer	n/a	n/a	any	NUMBER(38)
LongNVarChar	*n*	n/a	n/a	NCLOB
LongVarBinary	*n*	n/a	n/a	BLOB
LongVarChar	*n*	n/a	blank	CLOB
LongVarChar	*n*	n/a	Unicode	CLOB
NChar	n/a	any	n/a	Not supported
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	NVARCHAR2(*n*)[1]
Numeric	*p*	*s*	n/a	NUMBER(*p*,*s*)
Real	n/a	n/a	n/a	FLOAT(63)
SmallInt	n/a	n/a	any	NUMBER(38)
Time[2]	n/a	blank	blank	DATE
Time[2]	n/a	s	blank	DATE
Time[2]	n/a	blank	Microseconds	DATE
Time[2]	n/a	s	Microseconds	DATE
Timestamp[3]	n/a	blank	blank	DATE
Timestamp[3]	n/a	*s*	blank	DATE
Timestamp[3]	n/a	blank	Microseconds	DATE
Timestamp[3]	n/a	*s*	Microseconds	DATE
TinyInt	n/a	n/a	any	NUMBER(38)
VarBinary	*n*	n/a	n/a	LONG RAW
VarChar	*n*	n/a	blank	VARCHAR2(n)[5]

Table 3. Mapping of InfoSphere DataStage data types to Oracle data types (continued)

SQL type	Length	Scale	Extended	Oracle Column Definition
VarChar	*n*	n/a	Unicode	VARCHAR2(n)

Table notes:

1. When a column of InfoSphere DataStage Date type is used on the input link to write to an Oracle DATE or TIMESTAMP column, the hour, minute and second portions in the target value are set to midnight time. To change this default behavior you must use the *DS_DEFAULT_DATETIME_TIME* environment variable. Its format is HH:MI:SS where HH represents hours in 24-hour notation, MI represents minutes and SS represents seconds. When the environment variable is set, the stage uses the specified value for the default hour, minute and second portion for the target values. The environment variable does not provide an option to specify default fractional seconds when writing to Oracle TIMESTAMP. To be able to control fractional seconds you must use InfoSphere DataStage Time or Timestamp column on the link.
2. When a column of InfoSphere DataStage Time type is used on the input link to write to an Oracle DATE or TIMESTAMP column, the month, day and hour portions in the target value are set to current date (the exception is DRS Connector stage when the write mode is not Bulk load in which case zeros are used as default). To change this default behavior you must the *DS_DEFAULT_DATETIME_DATE* environment variable. Its format is YYYY-MM-DD where YYYY represents years, MM represents months and DD represents days. When the environment variable is set, the stage uses the specified value for the default year, month and day portion for the target values.
3. The Dynamic RDBMS stage uses Oracle SQL function `TO_DATE()` with format argument YYYY-MM-DD HH24:MI:SS when it generates `INSERT` statements for writing InfoSphere DataStage Timestamp values to Oracle DATE column. When it generates `SELECT` statements for reading Oracle DATE column into InfoSphere DataStage Date or Timestamp values it uses Oracle `TO_CHAR()` function to perform the conversion.

Informix data type support

When a job runs, the stage maps InfoSphere DataStage data types to Informix data types.

Table 4. Mapping of InfoSphere DataStage data types to Informix data types

SQL type	Length	Scale	Extended	Informix Column Definition
BigInt	n/a	n/a	any	Not supported
Binary	*n*	n/a	n/a	BYTE(*n*)
Bit	n/a	n/a	n/a	Not supported
Char	*n*	n/a	blank	CHAR(*n*)
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATE
Decimal	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Double	*p*	n/a	n/a	FLOAT(*p*)
Float	*p*	n/a	n/a	FLOAT(*p*)
Integer	n/a	n/a	any	INTEGER
LongNVarChar	*n*	n/a	n/a	Not supported
LongVarBinary	*n*	n/a	n/a	BYTE
LongVarChar	*n*	n/a	blank	TEXT
LongVarChar	*n*	n/a	Unicode	TEXT
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	NVARCHAR(*n*)
Numeric	*p*	*s*	n/a	Not supported

Table 4. Mapping of InfoSphere DataStage data types to Informix data types (continued)

SQL type	Length	Scale	Extended	Informix Column Definition
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	*s*	any	DATETIME
Timestamp	n/a	*s*	any	DATETIME
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	BYTE(*n*)
VarChar	*n*	n/a	blank	VARCHAR(*n*)
VarChar	*n*	n/a	Unicode	VARCHAR(*n*)

Additional notes:

- The `SERIAL` Informix data type is not supported.
- The extended Informix data types are not supported.

Microsoft SQL Server data type support

When a job runs, the stage maps InfoSphere DataStage data types to Microsoft SQL Server data types.

Table 5. Mapping of InfoSphere DataStage data types to Microsoft SQL Server data types

SQL type	Length	Scale	Extended	Microsoft SQL Server Column Definition
BigInt	n/a	n/a	any	Not supported
Binary	*n*	n/a	n/a	Not supported
Bit	n/a	n/a	n/a	Not Supported
Char	*n*	n/a	blank	CHAR(*n*)
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATETIME
Decimal	*p*	*s*	n/a	DECIMAL(*p,s*)
Double	*p*	n/a	n/a	FLOAT(*p*)
Float	*p*	n/a	n/a	FLOAT(*p*)
Integer	n/a	n/a	any	INTEGER
LongNVarChar	*n*	n/a	n/a	NTEXT
LongVarBinary	*n*	n/a	n/a	IMAGE
LongVarChar	*n*	n/a	any	TEXT
LongVarChar	*n*	n/a	Unicode	TEXT
NChar	*n*	n/a	n/a	NCHAR(*n*)
NVarChar	*n*	n/a	n/a	NVARCHAR(*n*)
Numeric	*p*	*s*	n/a	DECIMAL(*p,s*)
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	any	any	DATETIME

Table 5. Mapping of InfoSphere DataStage data types to Microsoft SQL Server data types (continued)

SQL type	Length	Scale	Extended	Microsoft SQL Server Column Definition
Timestamp	n/a	any	any	DATETIME
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	Not supported
VarChar	*n*	n/a	blank	VARCHAR(*n*)
VarChar	*n*	n/a	Unicode	VARCHAR(*n*)

Note: The Microsoft SQL Server `HierarchyId` data type is not supported

Sybase data type support

When a job runs, the stage maps InfoSphere DataStage data types to Sybase data types.

Table 6. Mapping of InfoSphere DataStage data types to Sybase data types

SQL type	Length	Scale	Extended	Sybase Column Definition
BigInt	n/a	n/a	any	Not supported
Binary	*n*	n/a	n/a	Not supported
Bit	n/a	n/a	n/a	Not supported
Char	*n*	n/a	blank	CHAR(*n*)
Char	*n*	n/a	Unicode	CHAR(*n*)
Date	n/a	n/a	n/a	DATETIME[1]
Decimal	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Double	*p*	n/a	n/a	FLOAT(*p*)
Float	*p*	n/a	n/a	FLOAT(*p*)[2]
Integer	n/a	n/a	any	INTEGER
LongNVarChar	*n*	n/a	n/a	TEXT
LongVarBinary	*n*	n/a	n/a	IMAGE
LongVarChar	*n*	n/a	blank	TEXT
LongVarChar	*n*	n/a	Unicode	TEXT
NChar	*n*	n/a	n/a	UNICHAR(*n*)
NVarChar	*n*	n/a	n/a	UNIVARCHAR(*n*)
Numeric	*p*	*s*	n/a	DECIMAL(*p*,*s*)
Real	n/a	n/a	n/a	REAL
SmallInt	n/a	n/a	any	SMALLINT
Time	n/a	any	any	DATETIME[3]
Timestamp	n/a	any	any	DATETIME
TinyInt	n/a	n/a	any	SMALLINT
VarBinary	*n*	n/a	n/a	Not supported
VarChar	*n*	n/a	blank	VARCHAR(*n*)
VarChar	*n*	n/a	Unicode	VARCHAR(*n*)

Table 6. Mapping of InfoSphere DataStage data types to Sybase data types (continued)

SQL type	Length	Scale	Extended	Sybase Column Definition

Table notes:

1. The time component of the Sybase DATETIME or SMALLDATETIME values is lost when converted to the InfoSphere DataStage Date data type. When writing InfoSphere DataStage Date values to a Sybase DATETIME or SMALLDATETIME column, the time component is set to midnight.
2. The InfoSphere DataStage Float data type has a maximum precision of 15 digits. Some loss of precision occurs when reading data from Sybase FLOAT(*p*) columns where *p* is greater than 15.
3. The date component of the Sybase DATETIME or SMALLDATETIME values is lost when converted to the InfoSphere DataStage Time data type. When writing InfoSphere DataStage Time values to a Sybase DATETIME or SMALLDATETIME column, the date component is set to the current date on the InfoSphere DataStage Server machine.

Considerations for processing large object (LOB) values

The stage can read and write large object (LOB) data types in the backend databases. The maximum size of the LOB values that are exchanged between the stage and the database needs to be carefully considered and the job design needs to be planned accordingly.

On the InfoSphere DataStage Server jobs the maximum size of the records on the link is limited by the system resources.

The Length attribute for any column on the link should be set to the smallest value large enough to accommodate the actual values that will be represented by that column when the job runs.

Note: In InfoSphere DataStage server jobs it is possible to use job parameter to dynamically specify the Length for the columns on the stage links. To do this, the *Description* attribute of the column needs to be set to a value in the following format:

```
param{length=#job_parameter_name#}
```

where *job_parameter_name* is the name of the job parameter through which the column length is specified when the job runs.

On the InfoSphere DataStage Parallel canvas in addition to the constraints dictated by the available system resources, the maximum size of records is also limited by the *APT_DEFAULT_TRANSPORT_BLOCK_SIZE* environment variable. This environment variable is a predefined InfoSphere DataStage environment variable that is automatically defined at the InfoSphere DataStage project level. The default value is `65536` bytes. To transfer records of a larger size, this variable needs to be increased accordingly.

For example, assume that the stage is configured to read data from a database table that contains one integer column and two LOB columns. And that the maximum size of the values in each of the LOB columns is 1 MB. The *APT_DEFAULT_TRANSPORT_BLOCK_SIZE* would then typically be set to `2097152` (2 MB) to accommodate for the values in the two LOB columns, plus another `65536` bytes to accommodate for the integer column and the internal record data in each record. So the actual value to specify would be `2162688`, which could then further be rounded to the value of `2200000`.

Transaction control

Dynamic relational database stages provide two levels of control for database transactions.

Size Specifies the number of rows per transaction

Isolation level
Specifies the type of transactions.

Transaction size

The transaction size is specified as the number of records to write in a transaction before committing the transaction and beginning the new one.

The transaction size can be specified when the stage is writing rows to the database except when it is writing the rows in bulk load mode.

If the number of rows in the last transaction in the job is smaller than the specified transaction size, the transaction is still committed.

Each time an array of rows is sent to the database, the stage checks if the number of rows that were written in the current transaction reached the specified transaction size. If the specified transaction size was reached, the stage commits the transaction and starts a new one. When writing rows to the database in array mode, the *Transaction size* should be set to a value that is a multiple of the specified *Array size*.

The transaction size of zero means that all rows that the stage receives on the input link for the duration of the job are written to the database in a single transaction.

In the Dynamic RDBMS stage the transaction size is specified in the **Transaction size** property on the **General** tab of the **Input** page. The default value is 1, which matches the default value of the **Array size** property for this stage.

Transaction isolation level

The transaction isolation level can be specified for the stage reading rows from the database or writing rows to the database. It specifies the type of transaction in terms of the locks that are acquired on the database objects affected by the transaction and that are kept on behalf of the stage for the duration of the transaction.

The dynamic relational stages provide a generic list of transaction isolation levels which are internally mapped to the corresponding isolation level supported by the backend database systems.

The supported transaction isolation levels in dynamic relational stages are:

Read Uncommitted
The transaction can see the changes made by other transactions that have not yet been committed. These are referred to as dirty reads.

Read Committed
The transaction cannot see the changes made by other transactions that have not yet been committed. Therefore dirty reads are not possible.

If the same statement is executed for the second time in the same transaction it can see the changes made to the rows that have been committed by other transaction after the current transaction started.

If the same statement is executed twice in the same transaction, it can retrieve rows with different values. These are referred to as non-repeatable reads.

This isolation mode also does not prevent the phantom reads. Phantom reads happen when the same statement is executed twice and the second statement fetches additional rows that have been inserted and committed by another transaction.

Repeatable Read
Similar to the Read Committed isolation level except that the changes committed by other transactions are not seen by the statements performed in the current transaction. This means that the non-repeatable reads are not possible.

Phantom reads are possible.

Serializable
Similar to the Repeatable Read isolation level except that phantom reads are prevented.

This is the most restrictive isolation level in terms of locks acquired on the database objects accessed by the transaction. It enforces the consistency within the transactions at the expense of their concurrency.

Not all databases support these same four levels of transaction isolation and in some cases they use different terminology for isolation levels. The dynamic relational stages map the value for the isolation level specified by the user to the closest supported value defined for the database type for which the stage was configured.

In the Dynamic RDBMS stage the transaction size is specified in the **Transaction Isolation** property on the **General** tab of the **Input** page. The default value is `Read Committed`.

Before SQL and After SQL statements

You can run custom SQL statements on the target database before and after data is processed.

Before SQL
Custom SQL statements are run when the job starts and before any data is processed by the stage.

After SQL
Custom SQL statements are run after all the data has been processed by the stage and just before ending the job.

If multiple statements are specified as Before SQL or After SQL statements, they need to be separated by a semicolon character.

The stage runs Before SQL statements in a one database transaction and After SQL statements in another. If a pair of semicolons is encountered in the statements, the stage interprets it as the user-defined commit point. It commits the current transaction and starts a new one.

To configure the Dynamic RDBMS stage to run **Before SQL** and **After SQL** statements you must specify the statements in the **Before** and **After** tabs under the **SQL** tab on the **Input** or **Output** page (depending on whether the stage is configured to read data from or write data to the database).

You can specify Before SQL and After SQL statements in a file located on the InfoSphere DataStage Server machine. To point the stage to this file, instead of typing in the actual statements in the stage you must type `FILE=`*`filepath`* or `{FILE}` *`filepath`*, where *filepath* is the fully-qualified path to the file with the actual SQL statements.

The stage rolls back the current transaction and fails the job when any statement in the Before SQL and After SQL lists of statements fails to run successfully. You can change this default behavior.

To cause the Dynamic RDBMS stage to continue processing statements when this happens, you must set the **Treat errors as non-fatal** property located on the **Before** and **After** tabs under the **SQL** tab on the **Input** or **Output** page (depending on whether the stage is configured to read data from or write data to the database).

Data errors

Data errors occur when a stage fails to write a particular row to the database.

When a Dynamic RDBMS stage fails to write a particular row to the database, it logs a warning and continues to process rows both for server and parallel jobs. To change the behavior, set the **Treat warning message as fatal error** property on the **General** tab on the **Input** page.

Rejecting bad records

For a server job, you can configure the stage to reject bad records.

For this functionality to work the stage must not be processing records in array mode. The Array size property must be set to 1.

To reject rows that arrive on an input link of the stage and that could not be written to the database, the link must be coming from a Transformer stage. In the Transformer stage there must be another output link defined and marked as reject link and configured to accept records that could not be processed by the database stage. The records are sent to the reject link and are delivered to the downstream stage which then processes or stores them. For example, a Sequential File stage can be used to store the rejected records to a file for later analysis.

Note: The reject links in Transformer stage can be configured to accept records that could not be processed by the database stage on another link, or that could not be written to another link due to a constraint defined for that link in the Transformer stage.

Handling $ and # characters

InfoSphere DataStage does not allow for the use of # and $ characters in the column names on the links.

These restrictions may not exist for the column names in the database tables. InfoSphere DataStage supports an environment variable that can be used to handle

this situation. The name of the environment variable is *DS_ENABLE_RESERVED_CHAR_CONVERT*, a predefined environment variable in InfoSphere DataStage projects. Its value can be modified at the InfoSphere DataStage project level, or it can be added to a particular job as job parameter and set at the job level. It can be set to `Yes` or `No`. The default value is `Yes`.

The value `Yes` replaces `__035__` sequence of characters in the specified table and column names with # character. It also replaces `__036__` sequence of characters with a single $ character.

When you use the stage in a server job, specify the database column names in the user-defined SQL statements in their original form, including any # and $ characters in them. The column names on the links have the restriction for using # and $ characters. They are represented in the SQL statement text by question marks (bind parameter markers). The question marks are bound to the columns on the link by position in a top-down fashion, so that way the names of the columns on the link do not need to be present in the SQL statement text.

For example, for an `UPDATE` statement the following text can be entered:

```
UPDATE table_name SET ##B$ = ? WHERE $A# = ?
```

The association of question marks with column names on the link is not affected by the names of the columns on the link. Instead the first column on the link marked as Key column is associated with the question mark bind parameter in the `WHERE` clause and the first column on the link not marked as Key column is associated with the question mark bind parameter in the SET clause.

In the Dynamic RDBMS stage, the *DS_ENABLE_RESERVED_CHAR_CONVERT* environment variable is supported by the following flavors:

- Informix
- Informix Bulk Load
- MSSQL Bulk Load
- Oracle OCI Bulk Load
- Sybase
- Sybase Bulk Load
- DB2 Bulk Load

The following flavors behave as if the *DS_ENABLE_RESERVED_CHAR_CONVERT* environment variable was set by default and automatically makes the conversion:

- Oracle OCI
- ODBC
- DB2

SQL meta tags

SQL meta tags are platform-independent SQL functions that are supported by the Dynamic Relational Database stages.

At run time, these tags are translated into native database-specific SQL functions of the backend database. They can be applied wherever you provide SQL statements:

- User-defined SQL statements for writing data to the database
- User-defined `SELECT` statement for reading data from the database

- Before SQL and After SQL statements
- WHERE and other clauses for automatically generated SELECT statement

The use of SQL meta tags is supported also for the SQL statements in a file to which the stage points.

The following list describes the available SQL meta tags and their usage:

%Abs(x)
Returns the absolute value of the numeric argument.

%Coalesce(expr1, expr2, ...)
Returns the first non-null argument provided to the function. The arguments are expressions of any type.

%Concat
Used as an operator for concatenating strings. At runtime this meta tag is replaced by the string concatenation mechanism appropriate for the specific relational database. For example, in IBM DB2 it is replaced with CONCAT function that for arguments takes the operands specified on the left and right sides of the %Concat meta tag, while in Sybase it is replaced with a "+" operator. This meta tag is supported with the same limitations as the native concatenation mechanism on the specific relational database. For example, some databases allow concatenating a string with a numeric value while other databases flag this as an error. The Dynamic Relational Database stages make no attempt to check or convert the data types of either of the operands.

%CurrentDateIn
Returns the current date and can be referenced in the user-defined INSERT statement or the WHERE clause of the user-defined SELECT, UPDATE or DELETE statement.

%CurrentDateTimeIn
Returns the current datetime (timestamp) and can be referenced in the user-defined INSERT statement or the WHERE clause of the user-defined SELECT, UPDATE or DELETE statement.

%CurrentDateTimeOut
Returns the current datetime (timestamp) in string format and can be referenced in the select list of the user-defined SELECT statement.

%CurrentTimeIn
Returns the current time and can be referenced in the user-defined INSERT statement or the WHERE clause of the user-defined SELECT, UPDATE or DELETE statement.

%CurrentTimeOut
Returns the current time in string format and can be referenced in the select list of the user-defined SELECT statement.

%DateAdd(date_from, add_days)
Adds the number of days specified in the **add_days** argument to the date specified in **date_from** argument and returns the result. The **add_days** argument can be a negative number.

%DateDiff(date_from, date_to)
Returns the difference in number of days between two dates (**date_from** and **date_to**) passed as arguments.

%DateIn(dt)

Converts the date value to a format suitable for use in conditional expressions of the `WHERE` clause in the user-specified `SELECT`, `UPDATE` or `DELETE` statement. or for passing date values in the `INSERT` statement parameters. The dt argument is a date value or a date string literal in the form YYYY-MM-DD.

%DateOut(dt)

Converts the date value **dt** to a string format suitable for use in the select list of the user-specified `SELECT` statement.

%DatePart(DTTM_Column)

Returns the date portion of the specified datetime column.

%DateTimeDiff(datetime_from,datetime_to)

Returns a time value representing the difference between two datetimes in minutes.

%DateTimeIn(dtt)

Converts the datetime (timestamp) value to a format suitable for use in conditional expressions of the `WHERE` clause in the user-specified SELECT, UPDATE or DELETE statement or for passing datetime values in the INSERT statement parameters. The **dtt** argument is a timestamp value or a timestamp string literal in the form YYYY-MM-DD-hh:mm:ss.ffffff.

%DateTimeOut(datetime_col)

Converts the datetime (timestamp) value to a string format suitable for use in the select list of the user-specified `SELECT` statement.

%DecDiv(a,b)

Returns a floating point number representing the value of **a** divided by **b** , where **a** and **b** are numeric expressions.

%DecMult(a,b)

Returns a floating pointer number representing the value of **a** multiplied by **b** , where **a** and **b** are numeric expressions.

%DTTM(date,time)

Combines the specified date value with the specified time value and returns the resulting datetime (timestamp) value.

%FullJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)

Produces a full outer join expression for the specified tables.

TableName1OrJoin is the name of the first table or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%LeftJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)

Produces a left outer join expression for the specified tables.

TableName1OrJoin is the name of the first able or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%Like("literal")
Returns the LIKE expression that can be used as condition in SELECT statements to look for values that contain the specified literal string value. It generates the following expression:

like 'literal%'
If the literal value contains '_' or '\%' the %Like meta tag resolves to the following:

like 'literal%' escape '\'

%LikeExact(fieldname,"literal")
Returns the expression that can be used as condition in SELECT statement to look for the specified literal values. It generates the following expression:

= 'literal'
If the literal ends with the '%' wildcard the generated expression is:

like 'literal' [escape '\']

%NumToChar(Number)
Converts the specified numeric value into a character value.

%RightJoin(TableName1OrJoin,[tableAlias1],tableName2OrJoin,[tableAlias2], joinCondition)
Produces a right outer join expression for the specified tables.

TableName1OrJoin is the name of the first able or nested join in the join.

tableAlias1 is an alias for the first table in the join and is optional.

tableName2OrJoin is the name of the second table or nested join in the join.

tableAlias2 is an alias for the second table in the join and is optional.

joinCondition is the expression describing the join condition.

%Round(expression,factor)
Rounds the specified numeric expression to the specified scale factor before or after the decimal point. If factor is a string literal, it can be rounded to a negative number.

%Substring(source_str,start,length)
Returns the substring of the specified source string at the specified position and length.

%TimeAdd(datetime,add-minutes)
Adds the specified number of minutes (can be negative) to the specified datetime (timestamp) value.

%TimeIn(tm)
Converts the time value to a format suitable for use in conditional expressions of the WHERE clause in the user-specified SELECT, UPDATE or DELETE statement or for passing date values in the INSERT statement parameters. The tm argument is a time value or a date string literal in the form hh:mm:ss.ffffff.

%TimeOut(time_col)
Converts the time value to a string format suitable for use in the select list of the user-specified SELECT statement.

%TimePart(DTTM_Column)
Returns the time portion of the specified datetime column.

%TrimSubstr(source_str,start,length)
Returns the substring of the specified source string at the specified position and length. Any trailing whitespace characters are trimmed from the result.

%Upper(charstring)
Converts the specified string to uppercase. Wildcard characters (like %) can be present in the argument.

Table aliases and table joins

Aliases for the tables can be specified as the second and fourth argument in the SQL for outer, right and left join SQL meta tags.

For example:

```
SELECT Table1Alias.x, Table1Alias.y
FROM %LeftJoin (Table1, Table1Alias, Table2, Table2Alias, Table1Alias.x = Table2Alias.x)
WHERE Table1Alias.y > 2
```

will typically be converted by the stage to following native SQL:

```
SELECT Table1Alias.x,Table1Alias.y
FROM Table1 Table1Alias LEFT OUTER JOIN Table2 Table2Alias
ON Table1Alias.x = Table2Alias.x
WHERE Table1Alias.y > 2
```

Nested outer joins

SQL meta tags for joins can be used recursively to produce nested outer joins.

For example:

```
SELECT Table1.x
FROM %LeftJoin (%LeftJoin (Table1,, Table2,, Table1.x=Table2.x),,
 Table3,, Table1.z = Table3.z AND Table2.q = Table3.q)
WHERE Table1.a = `xyz'
```

will typically be converted by the stage to following native SQL:

```
SELECT Table1.x
FROM Table1 LEFT OUTER JOIN Table2 ON TABLE1.x = Table2.x
LEFT OUTER JOIN Table3 on Table1.z = Table3.z AND Table2.q = Table3.q
WHERE Table1.a = `xyz'
```

Migration from the Dynamic RDBMS stage to the DRS Connector stage

Jobs with Dynamic RDBMS stages can be automatically migrated to the jobs that use DRS Connector stages.

The migration process is performed using the tool IBM InfoSphere Connector Migration Tool, which is available on the client tier machine of IBM InfoSphere Information Server. The tool is accessible from the Start menu, under **All programs** > **IBM InfoSphere Information Server** > **IBM InfoSphere Connector Migration Tool**.

Once the tool is started and connection is established to the InfoSphere DataStage project, the jobs with DRS plug-in stages are detected and can be selected for migration. Before performing the migration customize the migration preferences through the Preferences... main menu option. For example if migrating jobs with DRS plug-in stages that are connecting to Oracle databases, select the variant that

corresponds to the Oracle client version installed on the IBM InfoSphere DataStage Server machine (engine tier). This selection is available under the **Variant selection** tab in the **Preferences** dialog.

The following sections provide additional notes and troubleshooting information that is applicable to specific migration aspects and scenarios.

Migration options for Informix, Sybase and MS SQL Server database types

The DRS Connector stage does not support Informix, Sybase and Microsoft SQL Server database types.

When a Dynamic RDBMS stage configured for one of these database types is migrated to a DRS Connector stage, the **Database type** property in the DRS Connector stage is set to value `ODBC`.

In order for the migrated job to run, you must ensure that the ODBC data source definition (DSN) specified for the **Connection name** in the DRS Connector stage is defined on the system and configured to point to the same database to which the original Dynamic RDBMS stage was pointing. InfoSphere Information Server provides a set of ODBC drivers. Drivers for Informix, Sybase and MS SQL Server database types are also included.

If the Update action property in the original Dynamic RDBMS stage was set to `Bulk insert`, the **Write mode** in the target DRS Connector stage will be set to value `Insert`. The stage that originally used to write data to the database through the native bulk load interface provided by the backend database will be inserting the data using SQL INSERT statements instead.

Note: The Sybase and Microsoft SQL Server ODBC drivers included with Information Server support bulk write mode of operation. This option can be configured for the ODBC DSN definitions used by the DRS Connector stages that were migrated from the Dynamic RDBMS stages configured for Microsoft SQL Server or Sybase bulk load write mode.

Schema reconciliation messages in the job log

When a job with a Dynamic RDBMS stage is converted to the job with a DRS Connector stage, the new job can report schema reconciliation messages (of warning or informational severity) which have not been reported in the original job.

This will typically be the case for DRS Connector stages configured for the IBM DB2 or ODBC database types. For the DRS Connector stages configured for Oracle database type, the schema reconciliation messages will less likely be reported because in this case the DRS Connector will defer most of the schema reconciliation logic to the Oracle database.

A DRS Connector stage configured for DB2 or ODBC database type will report schema reconciliation messages if it determines that the definition of the columns on the link does not match the definition of the corresponding columns in the database table.

There are generally three approaches for eliminating the schema reconciliation messages:

- Modify column definitions on the link to match (as close as possible) the corresponding column definitions in the database table.
- Modify column definitions in the database table to match (as close as possible) the corresponding column definitions on the link.
- Define a message handler that demotes or suppresses the schema reconciliation messages. The message handler definition can be applied to all the jobs in the project or to an individual job.

Example 1

For example, if a parallel InfoSphere DataStage job contains a DRS Connector stage configured for IBM DB2 database type and the stage is used to insert data to a DB2 database table, the input link of the stage contains the following columns:

Table 7. Example 1: Input link contains the following columns

Name	Type	Length	Scale	Extended
C1	Decimal	8	4	(not set)
C2	NVarChar	10	(not set)	(not set)
C3	VarChar	10	(not set)	(not set)
C4	Float	(not set)	(not set)	(not set)
C5	Timestamp	(not set)	(not set)	Microseconds

The target table in the database contains the following columns:

Table 8. Example 1: Target table contains the following columns

Name	Type	Length	Scale
C1	DECIMAL	6	2
C2	VARCHAR	10	(n/a)
C3	VARCHAR	5	(n/a)
C4	INTEGER	(n/a)	(n/a)
C5	TIMESTAMP	(n/a)	3

The job runs and the following messages are reported in the job log:

```
#1
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C1. When writing column DECIMAL(8,4) into
database column DECIMAL(6,2), truncation, loss of precision or data
corruption can occur.
#2
Type: Warning
Message Id: IIS-CONN-DAAPI-00396
Message: DRS_Connector_1: Writing the WVARCHAR column C2 into
a VARCHAR database column can cause data loss or corruption due to
character set conversions.
#3
Type: Warning
Message Id: IIS-CONN-DAAPI-00393
Message: DRS_Connector_1: The length of WVARCHAR column C2
cannot be validated because the database column is VARCHAR and character
set conversion is involved. Inadequate column lengths can lead to
data truncation or unexpected errors.
#4
```

```
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C3. When writing column VARCHAR(min=0,max=10)
into database column VARCHAR(min=0,max=5), truncation, loss of precision
or data corruption can occur.
#5
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C4. When writing column SFLOAT into database
column INT32, truncation, loss of precision or data corruption can
occur.
#6
Type: Warning
Message Id: IIS-CONN-DAAPI-00398
Message: DRS_Connector_1: Schema reconciliation detected a
size mismatch for column C5. When writing column DATETIME(fraction=6)
into database column DATETIME(fraction=3), truncation, loss of precision
or data corruption can occur.
```

The job log can report additional warnings or fatal errors, depending on the actual data on the link and the type of the stage on the other side of the link.

Message #1 is reported because the Decimal(8,4) column C1 on the link has larger precision and scale than the corresponding DECIMAL(6,2) column C1 in the target table. To eliminate this warning, the Length and Scale attributes for the column C1 on the link should be set to values 6 and 2 (or smaller) so that they do not exceed the precision and scale of the column C1 in the target table.

Messages #2 and #3 are reported because the Unicode column C2 on the link is used to write to a VARCHAR column in the database. To eliminate the warnings the type of column C2 on the link should be changed from NVarChar to VarChar.

Message #4 is reported because the VarChar column C3 on the link has larger length than the corresponding VARCHAR column C3 in the target table. To eliminate this warning, the Length attribute for the column C3 on the link should be set to value 5 or smaller so that it doesn't exceed the length of the column C3 in the target table.

Message #5 is reported because the Float column C4 on the link is used to write to an INTEGER column C4 in the target table. The range of allowed values for the Float InfoSphere DataStage type is larger than the range of allowed values for the INTEGER DB2 type. To eliminate this warning, the type of the column C4 on the link should be changed from Float to Integer (or to SmallInt or TinyInt).

Message #6 is reported because the Timestamp column C5 on the link has larger fractional second precision (6) than the TIMESTAMP(3) column C5 in the target table. To eliminate this warning, the Extended attribute for the column C5 on the link should cleared. The Scale attribute should be cleared as well or set to the value 3 or smaller so that it doesn't exceed the fractional second precision of the column C5 in the target table.

An alternative approach for eliminating messages #1 to #6 would be to modify column definitions in the target table so that they are in agreement with the column definitions on the link.

Yet another approach would be to define message handler for the job and demote the messages with the corresponding message identifiers from Warning severity to

Informational severity. For the example used here, the message identifiers that would need to be demoted are: IIS-CONN-DAAPI-00398 (for messages #1, #4, #5 and #6), IIS-CONN-DAAPI-00396 (for message #2) and IIS-CONN-DAAPI-00393 (for message #3). Note that demoting or suppressing schema reconciliation messages reported by the DRS Connector stage will have no effect on any related warning and error messages logged by the InfoSphere DataStage framework and other stages in the job.

Example 2

In this example a parallel InfoSphere DataStage job contains a DRS Connector stage configured for DB2 database type. The stage is used to read data from a DB2 database table. The output link of the stage contains the following columns:

Table 9. Example 2: Output link contains the following columns

Name	Type	Length	Scale	Nullable	Extended
C1	Decimal	6	2	No	(n/a)
C2	VarChar	10	(n/a)	No	(not set)
C3	VarChar	3	(n/a)	No	(not set)
C4	Float	(n/a)	(n/a)	No	(n/a)
C5	Date	(n/a)	(n/a)	No	(n/a)

The source table in the database contains the following columns:

Table 10. Example 2: Source table contains the following columns

Nsme	Type	Length	Scale	Nullable
C1	DECIMAL	8	4	No
C2	VARGRAPHIC	10	(n/a)	No
C3	VARCHAR	5	(n/a)	Yes
C4	FLOAT	(n/a)	(n/a)	No
C5	TIMESTAMP	(n/a)	3	No

The job runs and the following messages are reported in the job log:

```
#1
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C1. When reading database column DECIMAL(8,4)
into column DECIMAL(6,2), truncation, loss of precision or data corruption
can occur.
#2
Type: Warning
Message Id: IIS-CONN-DAAPI-00397
Message: DRS_Connector_0: Reading the WVARCHAR database column
C2 into a VARCHAR column can cause data loss or corruption due to
character set conversions.
#3
Type: Warning
Message Id: IIS-CONN-DAAPI-00393
Message: DRS_Connector_0: The length of VARCHAR column C2 cannot
be validated because the database column is WVARCHAR and character
set conversion is involved. Inadequate column lengths can lead to
data truncation or unexpected errors.
#4
```

```
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C3. When reading database column VARCHAR(min=0,max=5)
into column VARCHAR(min=0,max=3), truncation, loss of precision or
data corruption can occur.
#5
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C4. When reading database column DFLOAT into column
SFLOAT, truncation, loss of precision or data corruption can occur.
#6
Type: Warning
Message Id: IIS-CONN-DAAPI-00399
Message: DRS_Connector_0: Schema reconciliation detected a size
mismatch for column C5. When reading database column DATETIME(fraction=3)
into column DATE, truncation, loss of precision or data corruption
can occur.
#7
Type: Info
Message Id: IIS-CONN-DAAPI-00057
Message: DRS_Connector_0: Modified field: C5, attribute: IS_NULLABLE.
Design-time value: 0. External value: 1
```

The message #1 is reported because the DECIMAL(8,4) column C1 in the database has larger precision and scale than the corresponding Decimal(6,2) column C1 on the link. To eliminate this warning, the Length and Scale attributes for the column C1 on the link should be set to values 8 and 4 (or larger) so that they are not smaller than the precision and scale of the column C1 in the source table.

The messages #2 and #3 are reported because the VarChar column C2 on the link is used to read data from the VARGRAPHIC column C2 in the database. To eliminate the warnings the type of the column C2 on the link should be changed from VarChar to NVarChar (or Extended attribute for this column should be set to value Unicode).

The message #4 is reported because the VARCHAR(5) column C3 in the database has larger length than the corresponding VarChar(3) column C3 on the link. To eliminate this warning, the Length attribute for the column C3 on the link should be set to value 5 (or larger) so that it is not smaller than the length of the column C3 in the source table.

The message #5 is reported because the Float column C4 on the link is used to read data from the FLOAT column C4 in the target table. The range of allowed values for the FLOAT DB2 type is larger than the range of allowed values for the Float InfoSphere DataStage type. To eliminate this warning, the type of the column C4 on the link should be changed from Float to Double.

The message #6 is reported because the Date column C5 on the link does not support hours, minutes, seconds and fractional seconds and this column is used to read data from the TIMESTAMP(3) column C5 in the source table. To eliminate this warning, the type of the column C5 on the link should changed from Date to Timestamp. Also the Scale attribute should be set to value 3 or larger or the Extended attribute should be set to Microseconds. That way the fractional second precision for the column C5 on the link will not be smaller than the fractional second precision for the column C5 in the source table.

The message #7 is of informational severity and is reported because the column C5 in the source table supports NULL values, and the column C5 on the link does not. To eliminate this message, it is necessary to change the Nullable attribute for column C5 on the link from No to Yes.

An alternative approach for eliminating messages #1 to #7 would be to modify column definitions in the source table so that they are in agreement with the column definitions on the link.

Yet another approach would be to define message handler for the job and demote the messages with the corresponding message identifiers from Warning severity to Informational severity. For the example used here, the message identifiers that would need to be demoted are: IIS-CONN-DAAPI-00399 (messages #1, #4, #5 and #6), IIS-CONN-DAAPI-00397 (message #2) and IIS-CONN-DAAPI-00393 (message #3). The message #7 with message identifier IIS-CONN-DAAPI-00057 is already at the Informational severity so the message handler cannot be configured to further demote it. However it can be configured to suppress this message from the log. Note that demoting or suppressing schema reconciliation messages reported by the DRS Connector stage will have no effect on any related warning and error messages logged by the InfoSphere DataStage framework and other stages in the job.

Column aliases

When the DRS Connector stage is configured for the IBM DB2 and ODBC database types, the items that are specified in a `SELECT` statement must correspond to the column names on the output link of the stage.

For example, for SQL expressions in the select list the use of column aliases is required since the actual text of the expressions will not match the column names on the link. This applies to the native SQL expressions specific to the database type for which the stage is configured as well as to the SQL meta tags supported by the DRS Connector stage.

When a Dynamic RDBMS stage is migrated to a DRS Connector stage that is configured for IBM DB2 or ODBC database type and the migrated stage specifies `SELECT` statement which does not use aliases for the SQL expressions in the statement, you must add column aliases to the expressions so that they correspond to the column names on the link.

The DRS Connector stage configured for the Oracle database type and the Dynamic RDBMS stage allow the use of `SELECT` statements where items in the list do not match column names on the output link. This includes the case when SQL expressions are used without column aliases. Note though that the use of column aliases is highly recommended in this case as well.

For example, if the DRS Connector stage is configured to read data from an IBM DB2 or ODBC database type, and the output link of the stage contains columns C1, C2 and C3, the following statement with SQL meta tags will not work:

```
SELECT %ABS(C1), %MIN(C3), %MAX(C2) FROM TABLE1
```

For this statement to work, aliases would need to be provided for the SQL expressions used in the statement and those aliases would need to correspond to the column names on the link with which the SQL expressions should be associated.

For example the following statement would associate the expression %ABS(C1) with the column C1 on the link, the expression %MIN(C3) with the column C3 on the link and the expression %MAX(C2) with the column C2 on the link:

```
SELECT %ABS(C1) C1, %MIN(C3) C3, %MAX(C2) C2 FROM TABLE1
```

For the DRS Connector stage configured for Oracle database type and for the Dynamic RDBMS stage the statement without aliases will work and the stage will perform association between the expressions and the columns on the link in an ordered fashion: the expression %ABS(C1) will be associated with the column C1 on the link, the expression %MIN(C3) will be associated with the column C2 on the link and the expression %MAX(C2) will be associated with column C3 on the link. In some cases such an association strategy can be intended, but in other cases it may not be. To prevent the stage from performing association in the ordered fashion it is highly recommended to explicitly include column aliases that correspond to the names of the columns on the link, so that the stage can perform the association by name.

Handling TO_DATE() and TO_CHAR() SQL functions for the Oracle database type

The Dynamic RDBMS stage uses TO_DATE and TO_CHAR Oracle SQL functions to fetch data and write data for InfoSphere DataStage Date, Time and Timestamp data types.

For example, when this stage is configured to auto-generate SELECT statement and column C1 on the output link is defined as Date column, column C2 as Time column and column C3 as Timestamp column, they will appear in the select list of the generated SELECT statement respectively as:

```
TO_CHAR(C1, 'YYYY-MM-DD')
TO_CHAR(C2, 'HH24:MI:SS')
TO_CHAR(C3, 'YYYY-MM-DD HH24:MI:SS')
```

If the stage is configured to auto-generate an INSERT statement and column C1 on the input link is defined as Date column, column C2 as Time column and column C3 as Timestamp column, they will appear in the parameter list of the generated INSERT statement respectively as:

```
TO_DATE(:1, 'YYYY-MM-DD')
TO_DATE(:2, 'HH24:MI:SS')
TO_DATE(:3, 'YYYY-MM-DD HH24:MI:SS')
```

The user-defined SQL statements in Dynamic RDBMS stage also use TO_CHAR and TO_DATE SQL functions for Date, Time and Timestamp columns.

The DRS Connector stage does not require these functions for Date, Time and Timestamp columns. It exchanges data with the database using internal Oracle date and timestamp datatypes so no conversion to and from character data is involved.

When a Dynamic plug-in stage with TO_CHAR and TO_DATE functions in user-defined SQL statements is migrated to the DRS Connector stage, the jobs can fail due to this difference. The problem does not occur for migrated stages with auto-generated SQL statement, because the SQL statement that the DRS Connector stage will generate at runtime will not use TO_CHAR and TO_DATE functions like if those functions were used in the auto-generated statement generated at runtime by the Dynamic RDBMS stage in the original job. This is because auto-generated statements are always generated from scratch when the job runs and the

mechanism used by the Dynamic RDBMS stage does not have effect on the mechanism used by the DRS Connector stage.

The preferred way for resolving job failures dues to use of TO_CHAR and TO_DATE SQL functions in the user-generated SQL statement of the Dynamic RDBMS stage after migrating it to DRS Connector stage is to remove those functions from the statement.

For example if the user-defined `SELECT` statement contains expression TO_CHAR(C1, 'YYYY-MM-DD') in the select list, replace it with C1.

As another example, if the user-defined INSERT statement contains expression TO_DATE(:2, 'HH24-MI-SS') in the parameter list, replace it with :2.

Another way for resolving the problem is to modify types of the columns on the link that correspond to these SQL functions. The types should be changed to character type (such as Char or VarChar) with the length that corresponds to the date/time format used in the SQL function.

For example if the user-defined `SELECT` statement contains expression TO_CHAR(C1, 'YYYY-MM-DD') in the select list, change the data type for column C1 on the link from Date to Char(10) since 10 is the total number of character positions for date values in 'YYYY-MM-DD' format (for example 2010-09-27).

As another example, if the user-defined INSERT statement contains expression TO_DATE(:2, 'HH24-MI-SS') in the parameter list, replace the column on the link that corresponds to first parameter from Time to Char(10) since 10 is the total number of character positions for time values in 'HH24-MI-SS' format (for example 11:26:45).

Finally in cases when modifying the existing statements or column definitions on the links is not convenient, another option is to define *CC_ORA_BIND_DATETIME_AS_CHAR* environment variable in the InfoSphere DataStage project and set it to value TRUE. Alternatively it can be defined at the project level with the default value of FALSE and added to individual jobs as job parameter and overridden to value TRUE for them.

When *CC_ORA_BIND_DATETIME_AS_CHAR* environment variable is defined and set to TRUE, the DRS Connector stage uses character representation for Date and Timestamp values exchanged with the Oracle database, using the same date and time formats used by the Dynamic RDBMS stage. That way the connector mimics the behavior of the Dynamic RDBMS stage. The use of this environment variable is not recommended in general case since it forces the behavior in the DRS Connector which is added only for backward compatibility with Dynamic RDBMS stage and which involves data conversions that are normally not necessary and which in turn can have significant negative impact on the performance.

Oracle manual load connection settings

The DRS Connector stage must have a valid connection to an Oracle database to bulk load data manually. When the Dynamic RDBMS stage bulk loads data manually, it does not write data to the database table directly but instead produces a pair of control and data files. The control and data files can then be used by the SQL*Loader Oracle utility to load the data to the database.

When the Dynamic RDBMS stage performs manual load, it does not attempt to connect to the database. Consequently it is possible to provide invalid database connection information and invalid table name in the stage properties and the job will still complete successfully.

The DRS Connector stage also supports manual load mode for Oracle database type and like the Dynamic RDBMS stage it also creates a pair of control and data files. However, the DRS Connector will always attempt to connect to the database based on the connection property values specified for the stage. If the connection fails the job also fails.

When a job with the Dynamic RDBMS stage configured for Oracle manual load is migrated to use the DRS Connector stage, the job will fail if the database connection information in the stage is invalid. In order to be able to run the job successfully it is necessary to edit the connection properties in the stage and set them to valid Oracle connection values. The stage will then be able to connect to the database and the job will proceed.

Note: This requirement exists even if the connector does not use the connection to create control and data files. On the other hand, the table name specified in the stage does not have to match an existing table in the database.

Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible.

For information about the accessibility status of IBM products, see the IBM product accessibility information at http://www.ibm.com/able/product_accessibility/index.html.

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most web browsers. Because the information center uses XHTML, you can set display preferences in your browser. This also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Appendix B. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 11. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at www.ibm.com/account/
Training and certification	You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at http://www.ibm.com/training
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Appendix C. Accessing the product documentation

Documentation is provided in a variety of formats: in the online IBM Knowledge Center, in an optional locally installed information center, and as PDF books. You can access the online or locally installed help directly from the product client interfaces.

IBM Knowledge Center is the best place to find the most up-to-date information for InfoSphere Information Server. IBM Knowledge Center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open IBM Knowledge Center from the installed product or from a web browser.

Accessing IBM Knowledge Center

There are various ways to access the online documentation:

- Click the **Help** link in the upper right of the client interface.
- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in web clients.
- Type the address in a web browser, for example, when you are not logged in to the product.

  Enter the following address to access all versions of InfoSphere Information Server documentation:

  `http://www.ibm.com/support/knowledgecenter/SSZJPZ/`

  If you want to access a particular topic, specify the version number with the product identifier, the documentation plug-in name, and the topic path in the URL. For example, the URL for the 11.3 version of this topic is as follows. (The ⇒ symbol indicates a line continuation):

  ```
  http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/⇒
  com.ibm.swg.im.iis.common.doc/common/accessingiidoc.html
  ```

  Tip:

  The knowledge center has a short URL as well:

  `http://ibm.biz/knowctr`

  To specify a short URL to a specific product page, version, or topic, use a hash character (#) between the short URL and the product identifier. For example, the short URL to all the InfoSphere Information Server documentation is the following URL:

  `http://ibm.biz/knowctr#SSZJPZ/`

  And, the short URL to the topic above to create a slightly shorter URL is the following URL (The ⇒ symbol indicates a line continuation):

  ```
  http://ibm.biz/knowctr#SSZJPZ_11.3.0/com.ibm.swg.im.iis.common.doc/⇒
  common/accessingiidoc.html
  ```

Changing help links to refer to locally installed documentation

IBM Knowledge Center contains the most up-to-date version of the documentation. However, you can install a local version of the documentation as an information center and configure your help links to point to it. A local information center is useful if your enterprise does not provide access to the internet.

Use the installation instructions that come with the information center installation package to install it on the computer of your choice. After you install and start the information center, you can use the **iisAdmin** command on the services tier computer to change the documentation location that the product F1 and help links refer to. (The ⇒ symbol indicates a line continuation):

Windows

```
IS_install_path\ASBServer\bin\iisAdmin.bat -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

AIX® Linux

```
IS_install_path/ASBServer/bin/iisAdmin.sh -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

Where <host> is the name of the computer where the information center is installed and <port> is the port number for the information center. The default port number is 8888. For example, on a computer named server1.example.com that uses the default port, the URL value would be http://server1.example.com:8888/help/topic/.

Obtaining PDF and hardcopy documentation

- The PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1.
- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

Appendix D. Providing feedback on the product documentation

You can provide helpful feedback regarding IBM documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To provide a comment about a topic in IBM Knowledge Center that is hosted on the IBM website, sign in and add a comment by clicking **Add Comment** button at the bottom of the topic. Comments submitted this way are viewable by the public.
- To send a comment about the topic in IBM Knowledge Center to IBM that is not viewable by anyone else, sign in and click the **Feedback** link at the bottom of IBM Knowledge Center.
- Send your comments by using the online readers' comment form at www.ibm.com/software/awdtools/rcf/.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).

Notices and trademarks

This information was developed for products and services offered in the U.S.A. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session or persistent cookies. If a product or component is not listed, that product or component does not use cookies.

Table 12. Use of cookies by InfoSphere Information Server products and components

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
Any (part of InfoSphere Information Server installation)	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
Any (part of InfoSphere Information Server installation)	InfoSphere Metadata Asset Manager	• Session • Persistent	No personally identifiable information	• Session management • Authentication • Enhanced user usability • Single sign-on configuration	Cannot be disabled

Table 12. Use of cookies by InfoSphere Information Server products and components (continued)

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
InfoSphere DataStage	Big Data File stage	• Session • Persistent	• User name • Digital signature • Session ID	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere DataStage	XML stage	Session	Internal identifiers	• Session management • Authentication	Cannot be disabled
InfoSphere DataStage	IBM InfoSphere DataStage and QualityStage Operations Console	Session	No personally identifiable information	• Session management • Authentication	Cannot be disabled
InfoSphere Data Click	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Data Quality Console		Session	No personally identifiable information	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere QualityStage Standardization Rules Designer	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Information Governance Catalog		• Session • Persistent	• User name • Internal identifiers • State of the tree	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere Information Analyzer	Data Rules stage in the InfoSphere DataStage and QualityStage Designer client	Session	Session ID	Session management	Cannot be disabled

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS^{Link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS^{Link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.

Index

Printed in USA

SC19-4348-00