IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for Distributed Transactions

SC19-4255-00

IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for Distributed Transactions

SC19-4255-00

Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 79.

Contents

Chapter 1. Configuring access to data sources

To configure database connectivity, you must install database client libraries and include the path to these installed libraries in the library path environment variable. Apart from the library path, for certain database types you must set additional environment variables on the engine tier computer.

About this task

You must install the client libraries and include the directory that contains the database client libraries in the library path environment variables. For more information about setting environment variables, see "Setting the library path environment variable" on page 19.

Some 64-bit database client software installations include 32-bit version and 64-bit version of the client libraries. In this case, you must set the library path to point to the libraries that have the bit level that matches the bit level of the engine tier installation. Otherwise, when a job that uses the connector that requires the client libraries runs, an error is reported because the stage library is not able to load the database client libraries.

Procedure

1. Install database client libraries.
2. Configure access to data sources.

Configuring access to DB2 databases

To use the DB2 Connector stage in a job, you must configure DB2 environment variables and set the privileges for DB2 users. The DB2 connector connects to your databases by using the DB2 client on the InfoSphere® DataStage® nodes.

Before you begin

- Confirm that your system meets the system requirements for InfoSphere Information Server. Make sure that you are using a supported version of IBM® DB2®. For more information about system requirements, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Install the IBM DB2 client on all InfoSphere DataStage nodes, and make sure that the client is working correctly.
- Use the DB2 Configuration Assistant to test the DB2 client and server connection. If the DB2 client fails to connect to the DB2 server, jobs that use the DB2 Connector stage also fail.
- Catalog each database in the DB2 client.
- InfoSphere DataStage runs many processes for each job. Ensure that DB2 resources, configuration parameters, and manager configuration parameters are configured properly.
- Make sure that the **DB2_PMAP_COMPATIBILITY** registry variable is set to `ON` to indicate that the distribution map size remains 4,096 (4-KB) entries. Though DB2® version 9.7 database for Linux, UNIX, and Windows supports distribution map entries up to 32,768 (32 KB), the DB2 connector supports only 4-KB entries in distribution maps.

- If you plan to use the DB2 connector with DB2 for z/OS in jobs with sparsely arriving data (such as jobs that use the Change Data Capture stage), ensure that the idle timeout value set in the DB2 **IDTHTION** subsystem parameter is longer than the longest expected interval of inactivity for the DB2 Connector stages in the job.

Procedure

1. Grant the InfoSphere DataStage users SELECT privileges on the following tables:

Table 1. Required SELECT privileges

DB2 product	Tables that require SELECT privileges
DB2 Database for Linux, UNIX, and Windows	SYSCAT.COLUMNS SYSCAT.KEYCOLUSE SYSIBM.SYSDBAUTH SYSCAT.TABLES
DB2 for z/OS	**Note:** Before the data is loaded to data to DB2 for z/OS, make sure the user has the GRANT ALL access on SYSIBM.SYSPRINT: SYSIBM.SYSCOLUMNS SYSIBM.SYSINDEXES SYSIBM.SYSKEYCOLUSE SYSIBM.SYSKEYS SYSIBM.SYSPRINT SYSIBM.SYSTABLESPACE SYSIBM.SYSTABLES SYSIBM.SYSTABLEPART SYSIBM.SYSUSERAUTH
DB2 Database for Linux, UNIX, and Windows and z/OS	SYSIBM.SYSDUMMY1 SYSIBM.SYSVIEWS

2. On DB2 for z/OS, ensure that the DBA runs the DSNTIJSG installation job to install the **DSNUTILS** stored procedure. The **DSNUTILS** stored procedure is required to start the bulk loader on DB2 for z/OS. For more information, see http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z9.doc.inst/src/tpc/db2z_enabledb2supplstprocs.htm
3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.

   You must set the **DB2INSTANCE** environment variable even if the stage accesses the default DB2 instance. The instance that is specified in the **DB2INSTANCE** environment variable becomes the default instance that is used by the connector. If you want to use a DB2 instance other than the default, then enter the name of that instance in the **Instance** property of the DB2 connector in the **Properties** tab. The DB2 client installs the default instances.

Table 2. Default instance installed by the DB2 client

Operating System	DB2 Instance
Linux or UNIX	`db2inst1`
Microsoft Windows	`DB2`

4. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is shown in the table.

Table 3. Default path for DB2 client libraries

Operating System	DB2 Instance
Linux or UNIX	`/opt/IBM/db2/V9/lib64`
Microsoft Windows	`C:\IBM\SQLLIB\BIN`

5. Optional: If the globalization map name for the DB2 connector job does not match the current system locale on the engine tier, then set the **DB2CODEPAGE** environment variable to a codepage corresponding to the map name. The DB2 code page can also be set by using a DB2 registry variable.

Configuring the DB2 native ODBC driver on AIX

To configure and use the DB2 ODBC driver on AIX operating systems, you must modify the DB2 environment variables and the **ODBCINI** environment variable.

Before you begin

- Confirm that your system meets the system requirements and that you are using a supported version of IBM DB2 database systems. For more information, see System Requirements.

Procedure

1. Update the 64-bit ODBC driver for DB2 on AIX operating systems so that the DataDirect driver manager can load the ODBC driver:
   a. Open the `db2o.o` driver file, which is in the *$DB2_HOME*`/sqllib/lib64` directory.
   b. In the `db2o.o` driver file, add a link to the `db2o.o.so` file. For example, you can add the following link: `ln -s db2o.o db2o.o.so`
2. Add the following entry to the `$ODBCINI` file. The DataDirect driver manager uses the `DriverUnicodeType=1` entry to work with the ODBC driver for DB2.

```
****************************************************
****************************************************
Driver=/home/db2inst1/sqllib/lib64/db2o.o.so
DriverUnicodeType=1
```

3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.
4. Add the path to the directory that contains the client libraries to the library path environment variable.

Configuring access to Greenplum databases

You can configure access to a Greenplum database by configuring an ODBC data source definition (DSN) for Greenplum and by adding the directory location of the Greenplum parallel file distribution program (`gpfdist`) to the system path. The Greenplum Connector stage uses ODBC to connect and to execute statements and uses the `gpfdist` program to exchange data with the Greenplum database.

Configuring ODBC access to Greenplum databases on Linux and UNIX

To connect to a Greenplum database, you must first configure an ODBC data source definition (DSN) for the database by using the IBM Greenplum Wire Protocol ODBC driver.

Before you begin

- Ensure that the ODBC driver for Greenplum libraries is installed.
- Ensure that the path to the directory that contains the ODBC libraries are added to the library load path environment variable. On Linux and UNIX, the default path is `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The following table lists the name of the library path variable for the different operating systems.

Table 4. Library path variables

Operating system	Library path variable
HP-UX	LD_LIBRARY_PATH or SHLIB_PATH
IBM® AIX®	LIBPATH
Linux	LD_LIBRARY_PATH

- Ensure that the *ODBCINI* environment variable is set to point to the `.odbc.ini` file, which contains the ODBC DSN definitions.

  Note: The *ODBCINI* environment variable is automatically set in the `dsenv` script as part of the InfoSphere Information Server installation process.

Procedure

1. Add a new ODBC DSN definition for the Greenplum Wire Protocol to the `.odbc.ini` file.
2. Specify the Hostname and PortNumber to the host and port where Greenplum database server is running.
3. Specify the Database name to default database to use for connections using the new ODBC DSN.
4. Save the `.odbc.ini` file.

Example

```
[Greenplum_DEV_SERVER]
Driver=/opt/IBM/InformationServer/Server/branded_odbc/lib/VMgplm00.so
Description=DataDirect 7.0 Greenplum Wire Protocol
...
Database=gp_dev
HostName=gp_host
PortNumber=5432
```

where, *gp_dev* is the name of the Greenplum database the DSN connects to and *gp_host* is the host name where the Greenplum server resides.

For information about configuring other driver options, see the Greenplum Wire Protocol Driver chapter in the *DataDirect Connect Series for ODBC User's Guide*.

Configuring ODBC access to Greenplum databases on Windows

To connect to a Greenplum database, you must first configure an ODBC data source definition (DSN) for the database by using the IBM Greenplum Wire Protocol ODBC driver.

Before you begin

- Ensure that the ODBC driver for Greenplum libraries is installed.

Procedure

1. Start the Microsoft ODBC Data Source Administrator.
   - On a 32-bit Windows computer, click **Start > Control panel > Administrative Tools > Data Sources (ODBC)**
   - On a 64-bit Windows computer, navigate to `C:\Windows\SysWOW64\odbcad32.exe`.

     Note: On Windows, InfoSphere® Information Server is a 32-bit application. Even on a 64-bit Windows computer, the connector is running as a 32-bit application. Therefore you must use the 32-bit version of the ODBC Data Source Administrator as the Greenplum connector will not be able to locate DSN definitions created in the 64-bit ODBC Data Source Administrator.
2. On the System DSN page, click **Add**.
3. On the Create New Data Source page, select the IBM Greenplum Wire Protocol driver and click **Finish**. For information about configuring the driver options, see the Greenplum Wire Protocol Driver chapter in the *DataDirect Connect Series for ODBC User's Guide*.

Greenplum parallel file distribution program (gpfdist)

The Greenplum Connector stage exchanges data with the Greenplum server by using the Greenplum file distribution program, which is called `gpfdist`.

The `gpfdist` program runs on the database client, and it must be installed on the InfoSphere Information Server engine tier computer. For data to be transferred by using the `gpfdist` protocol, a network route must be present to enable bidirectional access by using an IP address and optionally the presence of a DNS server to facilitate the name resolution. The connector invokes a `gpfdist` process on every physical computer node and creates the external table. The host of the external table data is identified by the `fastname` entry in the parallel engine configuration file (`$APT_CONFIG_FILE`). In order for the connector to invoke `gpfdist` on each engine tier, the location of `gpfdist(%GPHOME_LOADERS%\bin)` must be in the system path. In addition, the location of `gpfdist` dependent libraries (`%GPHOME_LOADERS%\lib`) must be in the system library path. On Windows, the system environment variable *PATH* is updated in the Advanced system settings. On Linux, the *PATH* environment variable is updated in the `dsenv` script.

Note: On Windows, the Greenplum installer adds `%GPHOME_LOADERS%\bin` and *%GPHOME_LOADERS%\lib* to the *PATH* system environment variable. Verify that these directories are in the *PATH*.

For manually adding `%GPHOME_LOADERS%\bin` and *%GPHOME_LOADERS%\lib* to the *PATH* system environment variable see the topic on setting the library path environment variable.

Configuring access to Informix databases

In jobs that contain the Informix® CLI, Informix Load, Informix XPS Load, or Informix Enterprise stages, you must set environment variables so that the jobs can access Informix databases.

Configuring access for the Informix CLI, Informix Load, and Informix XPS Load stages

For the Informix CLI, Informix Load, and Informix XPS Load stages to access Informix databases, you must set the values of environment variables and add data source name (DSN) entries to the `.odbc.ini` file.

Before you begin

- Install the IBM Informix server.

Procedure

1. Set the `INFORMIXDIR` environment variable to point to the installation directory of the IBM Informix server.
2. Make sure that the `PATH` environment variable contains `$INFORMIXDIR/bin`.
3. Make sure that the library path environment variable contains`$INFORMIXDIR/lib:$INFORMIXDIR/lib/esql/$INFORMIXDIR/lib/cli`. The following is an example of environment variable settings on an AIX® system.

   ```
   INFORMIXDIR=/opt/informix/IDS
   LIBPATH=/opt/informix/IDS/lib:/opt/informix/IDS/lib/esql:/opt/informix/
   IDS/lib/cli
   PATH=/opt/informix/IDS/bin
   ```

4. For the Informix CLI stage, if the data source uses a translation DLL, you must add `INFORMIXDIR/lib/esql` to the shared library search path. If `INFORMIXDIR/lib/esql` is not added, a message that is related to loading translation DLL is logged in the IBM InfoSphere DataStage job log.

Example

The following example shows sample DSN entries for AIX, Solaris, HP, and Linux platforms.

```
[Informix]
Driver=/home/informix/csdk/lib/cli/iclit09b.so
Description=IBM INFORMIX ODBC DRIVER
Database=<stores_demo>
LogonID=<user_id>
Password=<password>
ServerName=<informixserver>
HostName=<informixhost>
Service=<online>
Protocol=ontlitcp EnableInsertCursors=0 GetDBListFromInformix=0
CLIENT_LOCALE=en_us.8859-1
DB_LOCALE=en_us.8859-1
CursorBehavior=0 CancelDetectInterval=0 TrimBlankFromIndexName=1
ApplicationUsingThreads=1 TRANSLATIONDLL=/home/informix/csdk/lib/esql/igo4a304.so
</online></informixhost></informixserver></password></userid></stores_demo>
```

Every Informix data source to which the IBM InfoSphere DataStage jobs connect must have an entry in the `.odbc.ini` file. You must specify values for the Database and Server name properties. The CLIENT_LOCALE and DB_LOCALE fields are optional. If you add login ID (UID) or password (PWD) properties, the User Name and Password properties can be left blank. The values that are specified for the

login ID (UID) and password (PWD) properties override the values that are specified for the User Name and Password properties in the .odbc.ini file.

Configuring the environment for Informix Enterprise stages

You must have the correct privileges and set the environment variables to use the Informix Enterprise stage. You must also have a valid account on the databases to which you connect.

Before you begin

- Install the client libraries. To run jobs with Informix XPS stages on AIX systems, install the Informix client SDK 3.5 (all modifications) with the Informix XPS server.
- Make sure that Informix XPS server is running.

Procedure

1. Set the INFORMIXDIR environment variable to point to the installation directory of the IBM Informix server.
2. Set the INFORMIXSERVER environment variable to point to coserver name of coserver 1 in sqlhosts. Make sure that the coserver is accessible from the node on which you invoke the IBM InfoSphere DataStage job.
3. Set the INFORMIXSQLHOSTS environment variable to point to the sql hosts path. For example, /disk6/informix/informix_runtime/etc/sqlhosts.
4. To run jobs with Informix XPS stages on AIX systems, set the **LIBPATH** environment variable as follows: LIBPATH=$APT_ORCHHOME/lib:$INFORMIXDIR/lib:`dirname $DSHOME`/branded_odbc/lib:$DSHOME/lib:$DSHOME/uvdlls:$DSHOME/java/jre/bin/classic:$DSHOME/java/jre/bin:$INFORMIXDIR/lib:$INFORMIXDIR/lib/cli:$INFORMIXDIR/lib/esql

Configuring access to JDBC data sources

Before you can use the JDBC connector, you must set up the driver configuration file. The connector uses this file to obtain information about the available JDBC drivers in your system.

Procedure

1. Create a driver configuration file named isjdbc.config with read permission enabled for all the users. The driver configuration file name is case sensitive.
2. Open a text editor and include the following two lines that specify the class path and driver Java classes:

   ```
   CLASSPATH=driver_classpath
   CLASS_NAMES=driver_class_names
   ```

   The value driver_classpath is the cumulative Java class path for the JDBC drivers that you plan to utilize through the connector. The value is specified as a semicolon separated list of fully-qualified directory paths, .jar file paths and .zip file paths. For example, if the driver that you plan to use is implemented as a .jar file, include the full path to that .jar file in the driver_classpath value. For more information about the class path requirements for your driver, see the driver documentation.

   The value driver_class_names is a semicolon separated list of fully-qualified driver class names implemented by the JDBC drivers that you plan to utilize through the connector. The driver classes are the Java classes in the drivers that implement the java.sql.Driver JDBC API interface. For more information about the driver class implemented by the driver, see your JDBC driver

documentation. Note that you do not need to provide this information for the drivers implemented as JAR files and based on JDBC 4.0 or later JDBC specification because in those cases the connector is able to automatically determine the driver class name from the driver JAR file.

3. Save the `isjdbc.config` file on the InfoSphere Information Server engine tier host under the directory `IS_HOME/Server/DSEngine` directory, where *IS_HOME* is the InfoSphere Information Server home directory. For example, the home directory might be `C:\IBM\InformationServer` on a Windows system or `/opt/IBM/InformationServer` on a Linux or UNIX system. If the engine tier in your InfoSphere Information Server installation consists of multiple hosts, this file must be available from the same location on all the hosts. You can make this file available from the same location on all the hosts, by configuring the `DSEngine` directory as a shared network directory.

Example

This example shows how you can set up the driver configuration file.

Assume the following details:

- You want to use the JDBC connector with the following two JDBC drivers: JDBC 4.0 driver, Driver A and JDBC 3.0 driver, Driver B.
- Driver A is implemented as the `/opt/productA/driverA.jar` file, and Driver B is implemented as the`/app/productB/driverBimpl.jar` file.
- For Driver A, you determine that the name of the driver Java class is `com.example.A.Driver`, and for Driver B, you determine that the driver Java class name is `com.example.DriverB`.

To set up the driver configuration file, complete the following actions:

1. Create the `isjdbc.config` file and enter the following two lines:

```
CLASSPATH=/opt/productA/driverA.jar;/app/productB/driverBimpl.jar
CLASS_NAMES=com.example.A.Driver;com.example.DriverB
```

   Note: The `com.example.A.Driver` value can be omitted from the `CLASS_NAMES` because the Driver A is a JDBC 4.0 driver, and for this driver the connector can automatically retrieve the driver class name from the driver JAR file.

2. Save the file on the InfoSphere Information Server engine tier host under the `IS_HOME/Server/DSEngine` directory, where *IS_HOME* is the InfoSphere Information Server home directory.

After making changes to the driver configuration file you do not need to restart the DataStage engine, ISF agents or WebSphere Application Server. The JDBC connector recognizes the changes that are made to this file the next time it is used to access JDBC data sources.

Configuring access to Microsoft SQL Server databases

To configure access to Microsoft SQL Server, you must set the **ODBCINI** environment variable. You must also ensure that the Microsoft SQL Server database is accessible from the Microsoft SQL Server client and test connectivity between the Microsoft SQL Server client and the Microsoft SQL Server databases.

Before you begin

- Install client libraries.

About this task

When you are connecting to a remote database, ensure that the database server is configured to allow remote connections over the TCP/IP protocol.

The Microsoft SQL Server Client cannot be installed on UNIX. Because of this, the InfoSphere DataStage Dynamic RDBMS plug-in stage on UNIX cannot use the Bulk Insert mode operation when the stage is configured for Microsoft SQL Server database type. The DRS plug-in stage on Windows uses the Microsoft OLE DB API for Bulk Load operations and the API is not available on UNIX. When the DRS plug-in stage is configured for Microsoft SQL Server database on UNIX, the database type for the stage is automatically switched to ODBC.

For more information about configuring access to SQL Server InfoSphere DataStage, see the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for Microsoft SQL Server and OLE DB Data*.

Procedure

1. On UNIX or Linux, set the **ODBCINI** environment variable to point to the `.odbc.ini` file in which the Microsoft SQL Server connection definitions are created.
2. From the Microsoft SQL Server driver on Windows, test if the Microsoft SQL Server database is ready to receive incoming connections:
   a. In the Microsoft SQL Server DSN configuration window, specify the connection information for the Microsoft SQL Server database.
   b. Click **Finish**.
   c. Click **Test Data Source**.
3. On UNIX, to test if the SQL server database is ready to receive incoming connections:
   a. Open the UNIX command-line utility.
   b. Run the DataDirect ODBC example application from the *installation_directory*`/example` directory, where *installation_directory* is the path to the DataDirect ODBC example application installation directory.

```
[root@RH2011 example]# ./example
./example DataDirect Technologies, Inc. ODBC Example Application.

Enter the data source name : mysqlserver

Enter the user name        : username

Enter the password         : passoword

Enter SQL statements (Press ENTER to QUIT)
SQL>
Exiting from the Example ODBC program
[root@RH2011 example]# pwd
installation_directory/example
[root@RH2011 example]#
```

Configuring access to Netezza databases

To configure access to Netezza databases, you must install and configure the Netezza ODBC driver and create the data source.

Configuring access to Netezza databases on Linux and UNIX

To configure access to Netezza databases, you must specify parameters in the `.odbcinst.ini` file to configure the Netezza ODBC driver and also modify the `.odbc.ini` file to configure the data sources.

Before you begin

- Install client libraries.

About this task

If an `.odbcinst.ini` configuration file exists, you can modify the same file. If there is no existing `.odbcinst.ini` configuration file, then you can use the `odbcinst.ini.sample` to create the `.odbcinst.ini` configuration file. In most scenarios, you can use the contents of the `odbcinst.ini.sample` file without any changes. However, in the following scenarios, you must change the configuration file:

- If your client system was configured for ODBC drivers other than the Netezza ODBC driver and you want to continue to use those ODBC drivers, do not modify the existing entries in the `.odbcinst.ini` file. Add an entry for the Netezza ODBC driver at the end of the existing contents of the `.odbcinst.ini` file.
- If the Netezza client software and a Netezza ODBC driver were installed on your client system, check if the Netezza ODBC driver is configured. If it is not, add an entry to the end of the existing contents of the `.odbcinst.ini` file.

If the `.odbc.ini` configuration file exists in your home directory (For example, /home/myname) check if it contains entries for the Netezza appliance data sources to access. If it does not, copy the contents of the `odbc.ini.sample` file to the end of your existing `.odbc.ini` configuration file. Do not modify any existing entries in the file.

If you are using the InfoSphere Information Server version of the `.odbc.ini` configuration file on Linux, create a symbolic link in the folder where the configuration file exists to make sure that the Netezza connector works correctly:

1. Log on as the InfoSphere DataStage administrator.
2. To change to the installation directory of InfoSphere Information Server, enter the command: `cd /opt/IBM/InformationServer/Server/DSEngine` .
3. To create a symbolic link, enter the command: `ln -s .odbc.ini odbc.ini`.

Procedure

1. Log in using your user ID and password.
2. Configure the Netezza ODBC driver:
   a. Copy the contents of the `/usr/local/nz/lib/odbcinst.ini.sample` file.
   b. Modify the configuration entries depending on your requirement. Consult your Netezza system administrator to check if you must modify any specific configuration entries for your installation.
   c. Save the file as `.odbcinst.ini`.
3. Configure the Netezza appliance data sources:
   a. Copy the contents of `odbc.ini.sample` file into your home directory (for example, `/home/myname`) and rename it `.odbc.ini`.

b. Optional: To add the Netezza data sources to an existing `.odbc.ini` file, add the lines after [NZSQL] from the sample file to the existing `.odbc.ini` file. In the [ODBC Data Sources] section, add `NZSQL = NetezzaSQL` to the list of data source names.

c. Save and close the file.

4. Set the environment variables:

```
export ODBCINI=path_to_odbc.ini_file
export NZ_ODBC_INI_PATH=location_of_odbc.ini_file
```

Note: If the Netezza entries were added to an existing `odbc.ini` file, set only the **NZ_ODBC_INI_PATH** variable.

5. To restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring access to Netezza databases on Microsoft Windows

If InfoSphere Information Server runs on the Microsoft Windows operating system, you must create and configure the data source after you install the Netezza ODBC driver.

Before you begin

- Install database client libraries.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator `C:\Windows\SysWOW64\odbcad32.exe`, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector fails to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
  - On 32-bit Windows, the 32-bit driver is installed in the `C:\Windows\System32` directory.
  - On 64-bit Windows, you can install both 32-bit and 64-bit drivers. The 32-bit driver is installed in the `C:\Windows\SysWOW64` directory and 64-bit version is installed in the `C:\Windows\System32` directory.

Procedure

1. Create the data source:
   a. Do one of the following actions depending on your operating system:
      - On a 32-bit Windows system, click **Start** > **Control panel** > **Administrative Tools** > **Data Sources (ODBC)**.
      - On a 64-bit Windows system, use Explorer to navigate to `C:\Windows\SysWOW64\odbcad32.exe`.

   b. On the System DSN page, click **Add**.

   c. On the Create New Data Source page, select **NetezzaSQL** as the driver to set up the data source for, and then click **Finish**.
2. Configure the ODBC driver:

a. On the Netezza ODBC Driver Setup page, specify details about the data source.
b. In the **Server** field, specify the host name or the IP address of the Netezza system to which the ODBC driver connects.
c. To test the connection, specify the username and password, and then click **Test Connection**.

Configuring access to ODBC data sources

There are no special installation requirements for the ODBC connector . However, Before you can use the ODBC connector in a job, you need to configure database drivers, driver managers, and data source names.

Database drivers

You must install and configure database drivers and at least one driver manager before you can use the ODBC connector.

Supported drivers are included with the installation of the product.

The following drivers have limitations or special configuration requirements when you use them with the ODBC connector:

IBM Text File Driver

- Supported data types. Only Numeric, Date, and VarChar are supported.
- Unsupported runtime data types. The following runtime data types are not supported:
  - Bit
  - Binary
  - LongNVarChar
  - LongVarBinary
  - LongVarChar
  - NChar
  - NVarChar
  - Time
  - Timestamp

SQL Server driver

- Large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

SQL Server native driver

- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

SQL Server wire driver

- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.
- View design-time data. To view design-time data that contains spaces, you must select the quoted identifiers check box on the **Advanced** tab of the driver setup window.

Supported data sources

Before the ODBC connector can open a table or file to read and write data, a connection must be defined to the data source that contains that table or file.

To be a supported data source, the ODBC connector must be able to perform read, write, and lookup SQL statements and to exchange data between external data sources and the IBM InfoSphere DataStage data sets. You must also define names for each data sources that are specific to the operating system, driver manager, and driver.

For the ODBC connector, the ODBC driver manager establishes the connection. To establish your data sources, the following requirements must be met:

- The driver and driver manager must be installed on the system where the connector is installed and running. The data source can be on a remote system. You can connect to multiple data sources.
- For parallel jobs, the driver and driver manager must be installed on every node the ODBC connector runs.

You can connect to only one ODBC driver manager at a time.

When you configure the driver manager and you are logged into the IBM InfoSphere DataStage and QualityStage® Designer, you can see a list of data source names in the **Data source** property by clicking the **Data source** button. If you are working in a design environment that is not connected to the server, you can type a value in the **Data source** property.

Configuring the ODBC driver and the ODBC data source name

To use ODBC data sources in a job, you must configure the ODBC driver and the ODBC data source name (DSN) definitions.

Before you begin

- Install client libraries.
- Test the connection to the ODBC data source.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator `C:\Windows\SysWOW64\odbcad32.exe`, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector cannot to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
- For more information, see the topics about configuring ODBC access in the configuring product modules section of the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.

Procedure

1. Configure the ODBC DSN definitions:

Table 5. Configuring the ODBC data source name definitions

Operating System	Procedure
UNIX or Linux	Set the ODBCINI environment variable to point to the `.odbc.ini` file. The`.odbc.ini` file contains the ODBC DSN definitions. **Note:** The ODBCINI environment variable is set in the `dsenv` script automatically as part of the InfoSphere Information Server installation process.
Microsoft Windows	The DSN definitions are managed by the ODBC driver manager application included with the operating system. The ODBC DSN definitions must be configured as System DSN definitions in the ODBC data source Administrator. The ODBCINI environment variable is not applicable on Microsoft Windows.

2. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is as follows:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`. On the Microsoft Windows, the ODBC driver manager library is provided by the operating system. The location of the ODBC driver manager is automatically included in the PATH environment variable.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The ODBC driver manager is included with InfoSphere Information Server
3. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring the ODBC data source in a parallel processing environment

You can configure the ODBC data source in a parallel processing environment with one conductor node and multiple player nodes.

Before you begin

- Install client libraries.

Procedure

1. Add the path to the directory that contains the client libraries to the library path environment variable.
2. Copy the following directory from the conductor node to all computer nodes:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`.
3. Copy the `.odbc.ini` file from the conductor node to all computer nodes.

4. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
cd Install_directory/ASBNode/bin
. ./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

Configuring access to Oracle databases

You can configure access to an Oracle database from the Oracle client system by setting environment variables and by updating Oracle configuration files such as `tnsnames.ora` and `sqlnet.ora`. For more information, see the Oracle product documentation.

Before you begin

- Install client libraries.
- Ensure that your system meets the system requirements and that you have a supported version of the Oracle client and Oracle server. For system requirement information, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Ensure that the Oracle client can access the Oracle database. To test the connectivity between the Oracle client and Oracle database server, you can use the Oracle SQL*Plus utility.

About this task

You can use the `dsenv` script to update the environment variables that are used to configure access to Oracle databases. If you use the script, you must restart the server engine and the ASB Agent after you update the environment variables.

Procedure

1. Set either the **ORACLE_HOME** or the **TNS_ADMIN** environment variable so that the Oracle connector is able to access the Oracle configuration file, `tnsnames.ora`.
   - If the **ORACLE_HOME** environment variable is specified, then the `tnsnames.ora` file must be in the *$ORACLE_HOME*`/network/admin` directory.
   - If the **TNS_ADMIN** environment variable is specified, then the `tnsnames.ora` file must be in the *$TNS_ADMIN* directory.
   - If both environment variables are specified, then the **TNS_ADMIN** environment variable takes precedence.
   - Setting these environment variables is not mandatory. However, if one or both environment variables are not specified, then you cannot select a connect descriptor name to define the connection to the Oracle database. Instead, when you define the connection, you must provide the complete connect descriptor definition or specify an Oracle Easy Connect string.

   Note: If you use the Oracle Basic Instant Client or the Basic Lite Instant Client, the `tnsnames.ora` file is not automatically created for you. You must manually create the file and save it to a directory. Then specify the location of the file in the **TNS_ADMIN** environment variable. For information about creating the `tnsnames.ora` file manually, see the Oracle documentation.

2. Optional: Set the library path environment variable to include the directory where the Oracle client libraries are located. The default location for client libraries are as follows:
   - On 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 the path would be `C:\app\username\product\11.2.0\dbhome_1\BIN`.
   - On Linux or UNIX, `u01/app/oracle/product/11.2.0/client_1`
3. Set the **NLS_LANG** environment variable to a value that is compatible with the NLS map name that is specified for the job. The default value for the **NLS_LANG** environment variable is `AMERICAN_AMERICA.US7ASCII`.

   The Oracle client assumes that the data that is exchanged with the stage is encoded according to the **NLS_LANG** setting. However, the data might be encoded according to the NLS map name setting. If the **NLS_LANG** setting and the NLS map name setting are not compatible, data might be corrupted, and invalid values might be stored in the database or retrieved from the database. Ensure that you synchronize the **NLS_LANG** environment variable and NLS map name values that are used for the job.

   On Microsoft Windows installations, if the **NLS_LANG** environment variable is not set, the Oracle client uses the value from the Windows registry.

Configuring access to Sybase databases

To configure access to Sybase databases, you set environment variables on the engine tier computer and specify database information in the interfaces file.

Before you begin

- Install and configure the SQL Server or Sybase client software. The BCPLoad stage uses the BCP API in the DBLIB/CTLIB and NetLIB client libraries. You must ensure that these components are installed on the InfoSphere DataStage server that set up as a client to the SQL Server DBMS. For more information, see DBMS documentation.
- Make sure that the Sybase database server is accessible from the Sybase client.
- Create table in the database on the SQL Server.
- Make sure that the database is registered on the Sybase client.

Procedure

1. Set the **SYBASE** environment variable to point to the Sybase installation directory. For example, `export SYBASE=/disk3/Sybase`.
2. Set the **SYBASE_OCS** environment variable to point to the Sybase Open Client directory. For example, `export SYBASE_OCS=OCS-12_5`. This value indicates the version and release of the Open Client product.
3. Specify the database name, host system name or IP address, and port number in the interfaces file (for example, `sql.ini`) in the `$SYBASE` directory.
4. Set the **DSQUERY** environment variable to the name of the Sybase database server to connect to by default when the server name is not specified in the connection request. If the environment variable is not set, the default value SYBASE is used.
5. Set the PATH and library path environment variable to point to the directory that contains the client libraries.

- On 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.
- On Linux and UNIX, $SYBASE/$SYBASE_OCS/lib.

Note: Make sure that $SYBASE/$SYBASE_OCS/bin is displayed first in the PATH environment variable.

6. For the BCPLoad stages, configure the database for the fast copy (bulk load) option, by using a stored procedure. When this option is used, data is loaded without recording every insert in a log file. For more information about using stored procedures, see *Using Stored Procedures*.
7. Get login privileges to Sybase by using a valid Sybase user name and corresponding password, server name, and database. These must be recognized by Sybase before you attempt to access it.
8. Use the dsedit utility that is provided with the Sybase Open Client to configure connection to the Sybase database.
9. Test the connectivity to Sybase database outside of InfoSphere DataStage by using tools such as $SYBASE/$SYBASE_OCS/bin/isql tool in Sybase Open Client.
10. Complete the following steps to access Sybase databases with NLS in Sybase enterprise stages.
    a. Create a database and configure the language of your choice. For example, create database <<database path>> COLLATION 932JPN for a Japanese (shift_jis) database.
    b. Install the IBM InfoSphere DataStage server in that particular language, for example, Japanese (shift_jis).
    c. To set the language for InfoSphere DataStage client, use the **NLS** tab in job properties to select the language.
    d. Make sure that the selected language is set as default in the operating system of the system on which the InfoSphere DataStage client is installed.

Permissions required to access Sybase databases

To complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases, you require specific privileges on the table.

Table 6. Permissions required to access Sybase databases.

You require write, read, upsert and lookup privileges on the table to complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases.

Operation	Options	Sys partition (only for Sybase ASE)	sys objects	SELECT privilege on table	INSERT privilege on table	Delete table	Create table
Write	Create/ Replace	Yes	Yes	No	No	No	Yes
Write	Append	Yes	Yes	Yes	Yes	No	No
Write	Truncate	Yes	Yes	Yes	Yes	Yes	No
Read	All	No	Yes	Yes	No	No	No

Table 6. Permissions required to access Sybase databases (continued).

You require write, read, upsert and lookup privileges on the table to complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases.

Operation	Options	Sys partition (only for Sybase ASE)	sys objects	SELECT privilege on table	INSERT privilege on table	Delete table	Create table
Upsert	Update/ Insert	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.	• UPDATE privilege on the table that you want to update. • INSERT privilege on the table into which you want to insert records.
Look up	All	No	Yes	Yes	No	No	No

Configuring access to Teradata databases

To configure access to Teradata databases, you must install Teradata tools and Teradata transporters and set the environment variables.

Before you begin

Install database client libraries.

Procedure

1. Install Teradata tools and utilities on all nodes that run parallel jobs. For more information, see the installation instructions in the Teradata product documentation.
2. Install the Teradata Parallel Transporter. For more information, see the installation instructions in the Teradata product documentation.
3. Set the environment variables.

Table 7. Required Environment variables

Operating System	Environment variables

AIX	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`LIBPATH=$TWB_ROOT/lib:$LIBPATH`
	`NLSPATH=$TWB_ROOT/msg/%N`
	`export TWB_ROOT PATH LIBPATH NLSPATH`

Table 7. Required Environment variables (continued)

Operating System	Environment variables

HP-UX	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`SHLIB_PATH=$TWB_ROOT/lib:$SHLIB_PATH`
	`NLSPATH=$TWB_ROOT/msg/%N`
	`export TWB_ROOT PATH SHLIB_PATH NLSPATH`

Solaris	`TWB_ROOT=/usr/tbuild/08.01.00.02`
	`PATH=$TWB_ROOT/bin:$PATH`
	`LD_LIBRARY_PATH=$TWB_ROOT/lib:$LD_`
	`LIBRARY_PATH NLSPATH=$TWB_ROOT`
	`/msg/%N export TWB_ROOT PATH LD_`
	`LIBRARY_PATH NLSPATH`

Testing database connections by using the ISA Lite tool

After you establish connection to the databases, test the database connection by running the IBM Support Assistant (ISA) Lite for InfoSphere Information Server tool.

For more information about the ISA Lite tool, see the topic about installation verification and troubleshooting.

Setting the library path environment variable

To apply an environment variable to all jobs in a project, define the environment variable in the InfoSphere DataStage and QualityStage Administrator. The values that are 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.

About this task

For example, suppose that directory `/opt/branded_odbc/lib` is specified as the value for the library path environment variable at the project level. Directory `/opt/IBM/InformationServer/Server/branded_odbc/lib`, which contains the same libraries but in a different location is already in the library path that is defined at the operating system level or the `dsenv` script. In this case, the libraries from directory `/opt/IBM/InformationServer/Server/branded_odbc/lib` are loaded when the job runs because this directory appears before directory `/opt/branded_odbc/lib` in the values that are defined for the library path environment variable.

The name of the library path environment variable depends on your operating system.

Operating system	Library path environment variable
Microsoft Windows	**`PATH`**
HP-UX	**`SHLIB_PATH`**
IBM AIX	**`LIBPATH`**
Other supported Linux and UNIX operating systems, and HP-IA	**`LD_LIBRARY_PATH`**

On Linux or UNIX operating systems, the environment variables can be specified in the `dsenv` script. InfoSphere Information Server installations on Windows

operating system do not include the dsenv script.

Setting the library path environment variable in the dsenv file

On Linux or UNIX operating systems, you can specify the library path environment variables in the dsenv script. When environment variables are specified in the dsenv script, they apply to all InfoSphere DataStage projects that run under the InfoSphere Information Server engine.

Before you begin

Install the client libraries.

Procedure

1. Log in as the DataStage administrator user (dsadm if you installed with the default name).
2. Back up the *IS_install_path*/Server/DSEngine/dsenv script. *IS_install_path* is the InfoSphere Information Server installation directory (/opt/IBM/InformationServer if you installed with the default path).
3. Open the dsenv script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.
5. Set up your environment with the updated dsenv file.

   ```
   . ./dsenv
   ```

6. Restart the InfoSphere Information Server engine by entering the following commands:

   ```
   bin/uv –admin –stop
   bin/uv –admin –start
   ```

7. Assume root user privileges, directly with the **su** command or through the **sudo** command if the DataStage administrator user is part of the sudoers list.

   ```
   sudo su - root
   ```

8. Change to the ASB Agent home directory by entering the following commands:

   ```
   cd Install_directory/ASBNode/bin
   ```

9. Restart the ASB Agent processes by entering the following commands:

   ```
   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start
   ```

Results

After you restart the ASB Agent process, the InfoSphere Information Server services take approximately a minute to register the event.

Setting the library path environment variable in Windows

On the Windows operating system, both the library path and **PATH** environment variables are represented by the **PATH** system environment variable. For InfoSphere Information Server engine and ASB Agent processes to detect changes in the environment variables, the changes must be made at the system level and the InfoSphere Information Server engine must be restarted.

Before you begin

Install the client libraries.

Procedure

1. To edit the **PATH** system environment variable, click **Environment Variable** in **Advance System Settings**, and then select **PATH**.
2. Click **Edit**, then specify the path to the directory containing the client libraries.
3. Click OK.
4. Restart the InfoSphere Information Server engine.
5. Restart the ASB Agent processes.

Chapter 2. Preparing to use the Distributed Transaction stage

Before you can use the Distributed Transaction stage, you must prepare your system for configuration. If you want to run distributed transactions that follow the XA specification, you must then configure the XA resource managers for the IBM DB2 connector or the Oracle connector. You can skip the XA configuration if you configure the jobs that includes the Distributed Transaction stage for guaranteed delivery.

Preparing your system for configuration

Before you run the Distributed Transaction stage, you must install and configure IBM WebSphere MQ.

Procedure

1. Install IBMWebSphere® MQ Server on the same physical server as the InfoSphere Information Server engine tier computer. For more information about installing WebSphere MQ Server, see http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp.
2. Create a queue manager, or use the default queue manager that is created when WebSphere MQ is installed. For more information about creating queue manager, see http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.explorer.tutorials.doc/bi00256_.htm

Configuring WebSphere MQ with DB2 XA resources

To run jobs that include the Distributed Transaction stage, you must configure IBM WebSphere MQ Server to work with distributed transactions.

Configuring WebSphere MQ (Windows)

To configure IBM WebSphere MQ with IBM DB2 XA resources on Microsoft Windows, do the following:

Procedure

1. Set up MQ queues, if you are using MQ messaging in the Distributed Transaction stage. For example, you might want to set up the following queues:

   SOURCEQ
   A source queue that provides input to the initial WebSphere MQ Connector stage.

   WORKQ
   A work queue that holds the output messages from the WebSphere MQ Connector stage, so that the job can be restarted from an intermediate stage.

   REJECTQ
   A queue that holds the messages that are rejected by the Distributed Transaction stage.

   You can create MQ queues by running the MQSC command line utility for managing queue managers, or by using MQ Explorer. To run the MQSC command line utility, issue the following command:

```
runmqsc qmgrname
```

In this command, *qmgrname* is the name of the queue manager.

To define a local queue on the queue manager, use the MQSC command line utility to issue the following command:

```
define qlocal(queuename) defpsist(yes)
```

In this command, *queuename* is the name of the queue. To define the queue as persistent, specify `defpsist(yes)`. Messages on a persistent queue are preserved when the queue manager is restarted.

2. Create a DB2 switch load file.

   WebSphere MQ provides a sample makefile for creating the DB2 switch load file. The default location of the sample makefile and source files is:

   ```
   C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm\
   ```

   The name of the sample DB2 switch load file is `db2swit.dll`. By default, the file is created in:

   ```
   C:\Program Files\IBM\WebSphere MQ\exits
   ```

3. Add XA resource manager configuration information for DB2 to the queue manager. To complete this step:
   a. Start the MQ Explorer.
   b. Right-click the name of the queue manager that you are using, and select **Properties**.
   c. In the **Properties** menu, select XA resource managers.
   d. Click **Add** to add an XA resource manager.
   e. In the **Add XA Resource** menu, enter the **Name**, **SwitchFile**, **XAOpenString**, and **ThreadOfControl** values:

      Name The name that you choose as the logical name of the resource. Specify any meaningful name.

      SwitchFile
      The name of the DB2 switch load file that you created. For example: `db2swit.dll`. If you saved the DB2 switch load file to a location that is not specified in the PATH environment variable, then specify the full path to this file.

      XAOpenString
      A string of the following form:

      database`,`*user*`,`*password*`,toc=t`

      - *database* is the database that contains the tables that are being updated in your distributed transaction jobs.
      - *user* and *password* are the user ID and password to access the database.

      ThreadOfControl
      Select **Thread**.

   f. Click **OK**. If the following message is displayed, ignore it:

      Error applying changes from page "XA resource managers". Reason: An unexpected error (50002) has occurred. (AMQ4999)

   g. Close MQ Explorer.
   h. From the Windows command line, issue these commands to stop and start the queue manager:

```
endmqm -w queue-manager-name
strmqm queue-manager-name
```

The Windows login ID under which you issue this command must be no more than eight characters, and must be a member of the Windows user group mqm.

4. Change DB2 configuration parameters.
   a. Grant the user who starts the queue manager the authority to perform DB2 work. For example, to give the user ID mqm authority to connect to the SAMPLE database, issue these commands on a Windows command line:

      ```
      db2 connect to SAMPLE
      db2 grant connect on database to user mqm
      ```

   b. Issue this command on a Windows command line to specify the mqmax.dll as the DLL that DB2 uses to call the queue manager for dynamic registration.

      ```
      db2 update dbm cfg using TP_MON_NAME mqmax
      ```

   c. Restart DB2.

Configuring WebSphere MQ (Linux)

To configure IBM WebSphere MQ with IBM DB2 XA resources on Red Hat Enterprise Linux, do the following:

Procedure

1. Set up MQ queues, if you are using MQ messaging in the Distributed Transaction stage. For example, you might want to set up the following queues:

   SOURCEQ
   A source queue that provides input to the initial WebSphere MQ Connector stage.

   WORKQ
   A work queue that holds the output messages from the WebSphere MQ Connector stage, so that the job can be restarted from an intermediate stage.

   REJECTQ
   A queue that holds the messages that are rejected by the Distributed Transaction stage.

   You can create MQ queues by running the MQSC command line utility for managing queue managers, or by using MQ Explorer. To run the MQSC command line utility, issue the following command:

   ```
   runmqsc qmgrname
   ```

   In this command, *qmgrname* is the name of the queue manager.

   To define a local queue on the queue manager, use the MQSC command line utility to issue the following command:

   ```
   define qlocal(queuename) defpsist(yes)
   ```

   In this command, *queuename* is the name of the queue. To define the queue as persistent, specify `defpsist(yes)`. Messages on a persistent queue are preserved when the queue manager is restarted.

2. Create a DB2 switch load file.

   WebSphere MQ provides a sample makefile named `xaswit.mak` for creating the DB2 switch load file. The default location of the sample makefile and source files is `/opt/mqm/samp/xatm/`.

Edit the xaswit.mak file before you run it, to set the DB2LIBPATH variable for the version of DB2 that you are using. If you installed DB2 in the default directory, the path is /opt/IBM/db2/V9.1/lib.

The name of the sample DB2 switch load file is db2swit. By default, the generated switch load file is placed in the following location: /opt/mqm/exits.

3. Add XA resource manager configuration information for DB2 to the queue manager. To complete this step:
   a. Edit opt/mqm/qmgrs/*queue-manager-name*/qm.ini.
   b. Add a stanza like this:

```
XAResourceManager:
  Name=resname
  SwitchFile=/opt/mqm/exits/db2swit
  XAOpenString=database,user,password,toc=t
  ThreadOfControl=THREAD
```

   - *resname* is a name that you choose as the logical name of the resource.
   - *database* is the database instance that contains the tables that are being updated in your distributed transaction jobs.
   - *user* and *password* are the user ID and password to access the database.

   c. From the Linux command line, issue these commands to stop and start the queue manager:

```
endmqm -w queue-manager-name
strmqm queue-manager-name
```

   The Linux login ID under which you issue this command must be no more than eight characters, and must be a member of the group mqm.
4. Change DB2 configuration parameters. Grant the user who starts the queue manager the authority to perform DB2 work. For example, to give the user ID mqm authority to connect to the SAMPLE database, issue these commands on a Linux command line:

```
db2 connect to SAMPLE
db2 grant connect on database to user mqm
```

Configuring WebSphere MQ (AIX)

To configure IBM WebSphere MQ with IBM DB2 XA resources on IBM AIX, do the following:

Procedure

1. Set up MQ queues, if you are using MQ messaging in the Distributed Transaction stage. For example, you might want to set up the following queues:

   SOURCEQ
   A source queue that provides input to the initial WebSphere MQ Connector stage.

   WORKQ
   A work queue that holds the output messages from the WebSphere MQ Connector stage, so that the job can be restarted from an intermediate stage.

   REJECTQ
   A queue that holds the messages that are rejected by the Distributed Transaction stage.

   You can create MQ queues by running the MQSC command line utility for managing queue managers, or by using MQ Explorer. To run the MQSC command line utility, issue the following command:

```
runmqsc qmgrname
```

In this command, *qmgrname* is the name of the queue manager.

To define a local queue on the queue manager, use the MQSC command line utility to issue the following command:

```
define qlocal(queuename) defpsist(yes)
```

In this command, *queuename* is the name of the queue. To define the queue as persistent, specify `defpsist(yes)`. Messages on a persistent queue are preserved when the queue manager is restarted.

2. Create a DB2 switch load file.

   WebSphere MQ provides a sample makefile named `xaswit.mak` for creating the DB2 switch load file. The default location of the sample makefile and source files is `/usr/mqm/samp/xatm/`.

   Edit the `xaswit.mak` file before you run it, to set the DB2LIBPATH32 variable for the version of DB2 that you are using. If you installed DB2 in the default directory, the path is `/usr/IBM/db2/V9.1/lib`.

   The name of the sample DB2 switch load file is `db2swit`. By default, the generated switch load file is placed in the following location: `/usr/mqm/exits`

3. Add XA resource manager configuration information for DB2 to the queue manager. To complete this step:

   a. Edit `usr/mqm/qmgrs/queue-manager-name/qm.ini`.

   b. Add a stanza like:

   ```
   XAResourceManager:
     Name=resname
     SwitchFile=/usr/mqm/exits/db2swit
     XAOpenString=database,user,password,toc=t
     ThreadOfControl=THREAD
   ```

   - *resname* is a name that you choose as the logical name of the resource.
   - *database* is the database that contains the tables that are being updated in your distributed transaction jobs.
   - *user* and *password* are the user ID and password to access the database.

   c. From the AIX command line, issue these commands to stop and start the queue manager:

   ```
   endmqm -w queue-manager-name
   strmqm queue-manager-name
   ```

   The AIX login ID under which you issue this command must be no more than eight characters, and must be a member of the mqm group.

4. Change DB2 configuration parameters. Grant the user who starts the queue manager the authority to perform DB2 work. For example, to give user ID mqm authority to connect to the SAMPLE database, issue these commands on a AIX command line:

   ```
   db2 connect to SAMPLE
   db2 grant connect on database to user mqm
   ```

Configuring WebSphere MQ with Oracle XA resources

To run jobs that include the Distributed Transaction stage, you must configure the Oracle connector as an XA resource.

Configuring WebSphere MQ with Oracle on Windows

To configure IBM WebSphere MQ with Oracle XA resources on Microsoft Windows, do the following

Procedure

1. Verify that the environment variable ORACLE_HOME is set.
2. Copy the Oracle switch load file `oraswit.dll` to `WebSphere Install folder\exits`. The default location is `C:\Program Files\IBM\WebSphere MQ\exits`.
3. Add XA resource manager configuration information for Oracle to the queue manager as follows:
   a. Start the WebSphere MQ Explorer.
   b. Right-click the name of the queue manager that you are using, and select **Properties** > **XA resource managers**.
   c. Click **Add** to add an XA resource manager.
   d. In the **Add XA Resource** menu, enter the **Name**, **SwitchFile**, **XAOpenString**, and **ThreadOfControl** values:

      Name The name that you choose as the logical name of the resource.

      SwitchFile
      The name of the Oracle switch load file. If you saved the Oracle switch load file to a location that is not specified in the PATH environment variable, then specify the full path to the file.

      XAOpenString
      A string of the form
      `ORACLE_XA<+required_fields...><+optional_fields...>`

      For example, `Oracle_XA+Acc=P/SCOTT/tiger+SesTm=20+SqlNet=orcl+DbgFl=0x07+LogDir=C:\xalogs\orcl+Threads=true+DB=myoraxa`

      You must specify the `DB` and `SqlNet` values to setup the Oracle Connector for use in DTS Jobs.

      ThreadOfControl
      Select **Thread**.
   e. Click **OK**. You can ignore the following message:

      Error applying changes from page "XA resource managers". Reason: An unexpected error (50002) has occurred. (AMQ4999)
   f. Close the WebSphere MQ Explorer.
   g. From the Windows command line, issue the following commands to stop and start the queue manager:

      ```
      endmqm -w queue-manager-name
      strmqm queue-manager-name
      ```

      The Windows login ID under which you issue this command must not be more than eight characters, and must be a member of the Windows user group **mqm**.
4. Set up WebSphere MQ queues, if you are using WebSphere MQ messaging in the Distributed Transaction stage. For example, you can setup the following queues:

   SOURCEQ
   A source queue that provides input to the initial WebSphere MQ Connector stage.

WORKQ
A work queue that holds the output messages from the WebSphere MQ Connector stage, so that the job can be restarted from an intermediate stage.

REJECTQ
A queue that holds the messages that are rejected by the Distributed Transaction stage.

5. To add queues by using the WebSphere MQ Explorer, right-click the Queues folder and select **New** > **Local Queue**.

Configuring the Oracle database

To run Distributed Transaction stage jobs for XA transactions between WebSphere MQ and Oracle, you must configure the Oracle database as a resource manager.

Procedure

1. Grant access to Oracle transaction management views. For more information about how to configure Oracle database, see the Oracle documentation about *Developing Applications with Oracle XA*.
2. Configure the Oracle database as the resource manager. When you configure the Oracle database as the resource manager, you generally specify values for the **DB** and **SqlNet** parameters in each XAOpenString entry. A value for the **DB** parameter is required only if a job writes to more than one Oracle resource manager. In that case, the parameter is used to differentiate between multiple Oracle resource managers.

   Record the value that you specify for the **SqlNet** parameter. If you define more than one Oracle resource manager, record the value for the **DB** parameter.

Setting up the Oracle connector for XA transactions

When you define the connection between the Oracle connector and the Oracle database, you specify the Oracle resource manager to write to.

Before you begin

- Configure an IBM WebSphere MQ queue manager as the transaction manager.
- Configure the Oracle database as the resource manager.

Procedure

Define the connection between the Oracle connector and the Oracle database. You must specify a value for the following fields:

XA database name
Enter the value that you specified for the **DB** parameter in the XAOpenString entry. This field is required only if you register more than one Oracle resource manager with the WebSphere MQ queue manager that the Distributed Transaction stage references.

Server Enter the value that you specified for the **SqlNet** parameter in the XAOpenString entry.

Chapter 3. Overview

You can use the Distributed Transaction stage to run distributed transactions with IBM WebSphere MQ as the transaction manager and IBM DB2, Oracle, or WebSphere MQ as the resource manager.

A transaction is a series of actions that are completed as a single operation. A transaction ends with a commit action that makes the changes permanent. If any of the changes cannot be committed, the transaction rolls back all of the changes.

A distributed transaction (also known as a global transaction) is a transaction that might span multiple data sources, such as one or more databases and a WebSphere MQ message queue. For the transaction to commit successfully, all of the individual data sources must commit successfully. If any resource cannot commit, the entire transaction is rolled back. For example, a distributed transaction might consist of a money transfer between two bank accounts that are on different databases. The transaction is committed only if the withdrawal from one account and the deposit into the other account are successfully completed.

The Distributed Transaction stage also supports guaranteed delivery. Guaranteed delivery is used for data sources that do not support the XA specification, such as ODBC or Teradata data sources. You can also run guaranteed delivery for data sources that support the XA specification if the connector supports guaranteed delivery. The following table shows the connectors that support distributed transactions and guaranteed delivery

Table 8. Connectors that support global transactions and guaranteed delivery

Connector	Support for distributed transactions (also known as global transactions)	Support for guaranteed delivery
IBM DB2 connector	Yes	No
Oracle connector	Yes	Yes
IBM WebSphere MQ connector	Yes	Yes
ODBC connector	No	Yes
Teradata connector	No	Yes

Distributed transaction processing model

The X/Open standard for distributed transactions defines a model for distributed transaction processing. In this model, a coordinating transaction manager manages how each data source processes a transaction, based on its knowledge of all the data sources that participate in the transaction. The data sources that normally manage their own transaction commit and recovery delegate this task to the transaction manager.

According to the X/Open standard, the distributed transaction processing model consists of the following components:

- An application program that defines transaction boundaries and specifies actions that constitute a transaction

- Resource managers, such as databases or file systems that provide access to shared data sources
- A transaction manager that assigns identifiers to transactions, monitors their progress, and manages transaction completion and failure recovery

The XA specification defines the two-phase commit protocol as the interface that is used for communication between a transaction manager and a resource manager. The two-phase commit protocol includes a prepare phase and a commit phase. During the prepare phase, all participants in the transaction must agree to complete the changes that are required by the transaction. If any of the participants reports a problem, the prepare phase fails, and the transaction is rolled back. If the prepare phase succeeds, the commit phase starts. During the commit phase, the transaction manager instructs all participants to commit the transaction.

In the Distributed Transaction stage, when WebSphere MQ is the transaction manager, data sources such as IBM DB2 and Oracle can participate in an XA transaction as resource managers.

For more information about the X/Open standard, see *Distributed Transaction Processing: The XA Specification*, which is available from http://opengroup.org.

Guaranteed delivery of data for connectors that do not follow the XA specification

You can use the Distributed Transaction stage to guarantee delivery of data when you use connectors such as the Teradata connector and ODBC connector that do not support distributed transactions.

With guaranteed delivery, no transaction manager coordinates the transactions. The transaction commit and recovery are managed by each data source. In this model, the Distributed Transaction stage guarantees that all data sources commit successfully when data is deleted from the source system. When a transaction commit fails, data remains in both the source and target systems. In transactions that follow the XA specification, data remains in only the source system or only the target system.

When the Distributed Transaction stage runs guaranteed delivery, the following actions are completed:

1. Target data sources are updated in a sequential order.
2. A transaction is committed in all target data sources in the same sequential order.
3. After all the target data sources are committed, the source WebSphere MQ messages are deleted from the source or the work queue.

Example 1

In a job the Distributed Transaction stage is configured as a target stage and the Teradata connector is used as a data source that does not follow the XA specification. When the job is run, the following actions occur:

1. A local transaction is started with the Teradata connector.
2. The target is updated.
3. If the updates are successful, the local transaction is committed when the end of wave is reached.

4. When the local transaction is successfully committed, the following actions occur:
   a. A WebSphere MQ transaction is started.
   b. The messages about the current transaction are deleted from the source or work queue.
   c. The WebSphere MQ transaction is committed.

In this example, if step 3 is successful and step 4 is unsuccessful, data can remain in both the source system and target system.

Example 2

If a data source that follows the XA specification, such as the DB2 connector, is added to the scenario in "Example 1" on page 32, and global transaction is enabled, the following actions occur:

1. A local transaction is started with the Teradata connector, which is a data source that does not support the XA transaction.
2. A distributed transaction is started.
3. The target that does not support the XA transaction is updated.
4. The DB2 data source is updated.
5. If the updates are successful, the local transaction is committed when the end of wave is reached.
6. When the local transaction is successfully committed, the following actions occur:
   a. The messages about the current transaction are deleted from the source or work queue.
   b. The distributed transaction is committed.

In this example, data can remain in both the source system and the target system that does not support XA transaction. If steps 3, 4, and 5 are successful and step 6 is unsuccessful, the data remains in the source or the work queue.

Typical job flow for a distributed transaction

You can use the Distributed Transaction stage to implement distributed transactions in IBM InfoSphere DataStage jobs. A typical job flow might include a WebSphere MQ Connector stage, one or more transformations stages, and a Distributed Transaction stage.

1. A WebSphere MQ Connector stage consumes source messages from an MQ message queue and moves the messages to a persistent MQ work queue. The connector copies the data, message ID, and other message header fields from the source to the target message. The WebSphere MQ Connector stage also sends the message data to an output link.
2. One or more stages process the message data.
3. The Distributed Transaction stage receives data from one or more links. Each input link to the stage represents output to a target database. The links provide the MQ message ID of the original source message, which is consumed from the work queue as part of the distributed transaction.

System configurations for Distributed Transaction stage jobs

Distributed Transaction stage jobs support system configurations in which MQ messages are ordered or unordered, and in which messages are related or unrelated.

You can design Distributed Transaction stage jobs that account for whether your MQ messages have either of the following characteristics:

ordering
The messages must be processed in the order in which they are written to the source queue. If the order must be maintained, the job cannot be run in parallel, because there is no coordination between processes on multiple nodes.

relationships
The source messages have some key field that indicates that they must be processed as a unit. All messages with a given key must be processed by the same node.

Configurations with no message ordering and no relationships between messages

If the order of processing of source messages is unimportant, and there is no relationship between messages, you can run Distributed Transaction stage jobs in parallel. The following figure illustrates a possible configuration with no ordering and no relationships between messages.

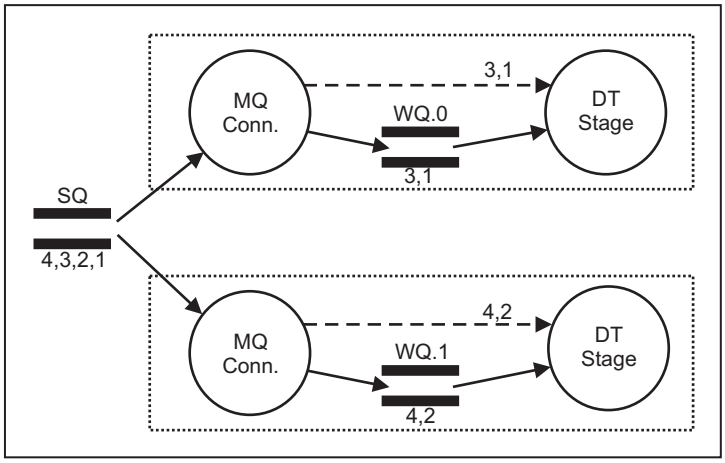


Figure 1. Distributed transaction configuration with no ordering and no relationships

This configuration contains two nodes. The source queue contains four messages. Messages 1 and 3 go to one node, and messages 2 and 4 go to the other node. Each node contains an IBM WebSphere MQ Connector stage, a work queue, and a Distributed Transaction stage. The WebSphere MQ Connector stages access a single source queue. The messages are distributed to the two nodes. To prevent the same source message from being read more than once, each WebSphere MQ Connector stage reads the messages destructively off the source queue. The messages are moved from the source queue to the work queues within local MQ transactions.

Because the messages are removed from the source queue, each node must have its own work queue. The work queues let you restart a job after a failure occurs. A

work queue for each node is required to prevent multiple MQ processes from reading the same message. Only a single work queue name value is specified in the WebSphere MQ Connector stage. When multiple parallel instances of the WebSphere MQ Connector stage are running, the connector automatically appends the dot and node number to the specified work queue name. For example, if the specified work queue name is WQ, then the work queue on the first node is WQ.1, and on the work queue on the second node is WQ.2.

Configurations with relationships among messages, but no message ordering

If messages can be processed in any order, but messages that are related to each other by a shared key value must be sent to the same node, you must use a single WebSphere MQ Connector stage configured to run sequentially combined with a hash partitioner. The following figure illustrates a possible configuration with relationships between messages, but no ordering.

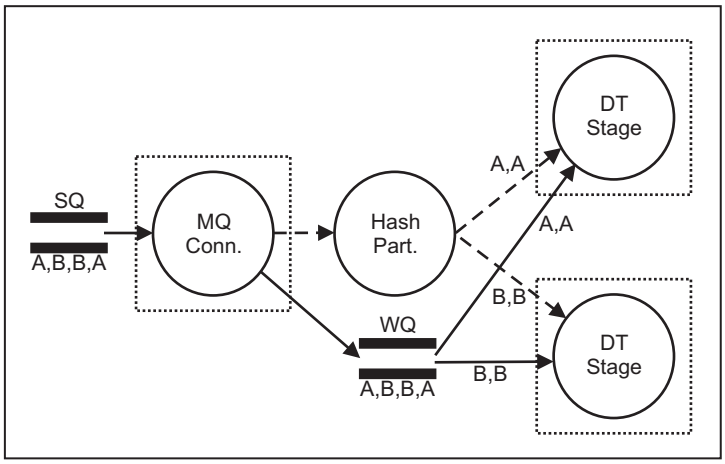


Figure 2. Distributed transaction configuration with relationships but no ordering among messages

This configuration has four source messages and two processing nodes. The A messages are related to each other by a shared key, and the B messages are related to each other by a shared key. Because the WebSphere MQ connector cannot determine which node is targeted for a specific message, you must use a single work queue, which directs the related messages to the same queue.

Configurations with message ordering

If messages must be processed in a particular order, run the job sequentially, and use a single WebSphere MQ Connector stage. The following figure illustrates a possible configuration with message ordering.

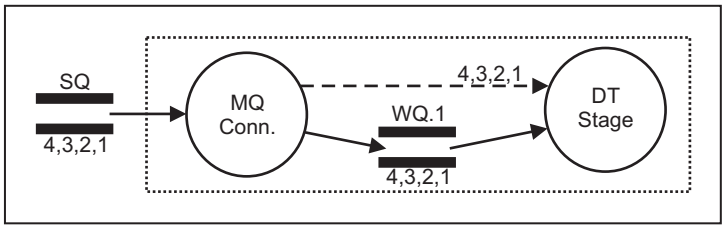


Figure 3. Distributed transaction configuration with message ordering

In this configuration, four source messages must be processed in order. A single processing node must be used; you cannot synchronize messages among multiple processing nodes. This configuration uses a work queue. However, because a single WebSphere MQ Connector stage is required, you do not have to use a work queue. If the job fails, you can restart it from the beginning without loss of data. The following figure illustrates an alternative configuration that does not include a work queue.

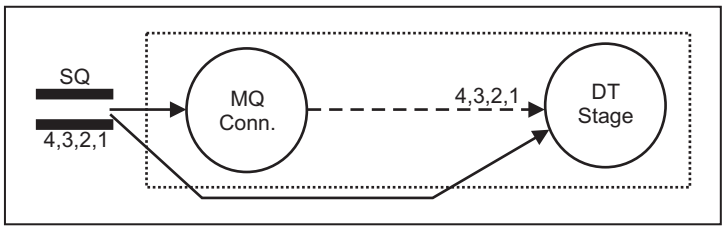


Figure 4. Distributed transaction configuration with message ordering and no work queue

In this configuration, four source messages must be processed in order. All four messages are read from the source queue and sent directly to the Distributed Transaction stage. If the job fails, the four messages are read from the source queue again, when the job is restarted.

Omission of the work queue can lead to better performance if there are a small number of messages. However, omission of a work queue can cause problems in the following situations:

- If multiple processes are writing to the source queue, the queue browser might miss a message if the PUT and COMMIT calls from these processes are interspersed in a certain order.
- If the processes that write to the source queue use message priorities, the queue browser does not see messages of a higher priority, because those messages move ahead of the current cursor position.

Chapter 4. Using the Distributed Transaction stage in your job design

With the Distributed Transaction stage, you can design jobs that access multiple resource managers so that work that is performed by those resource managers is committed or rolled back as a unit. You design the work that is in the distributed transaction as input links to the Distributed Transaction stage.

About this task

To set up a distributed transaction, perform the following tasks.

Setting up the distributed transaction job

You set up the distributed transaction job by adding the required stages to the parallel canvas.

About this task

To set up the distributed transaction job:

Procedure

1. In the InfoSphere DataStage and QualityStage Designer client, select **File** > **New**.
2. In the New window, select the **Parallel Job** icon, and click **OK**.
3. For jobs that require WebSphere MQ message queues, add a WebSphere MQ Connector stage to the job:
   a. In the Palette window, select the **Real Time** category.
   b. Locate **WebSphere MQ** in the list of available items, and click the down arrow to view the available connectors.
   c. Drag the **WebSphere MQ Connector** stage to the parallel canvas.
4. Add a Distributed Transaction stage to the job:
   a. In the Palette window, select the **Database** category.
   b. Locate **Distributed Transaction** in the list of available items.
   c. Drag the **Distributed Transaction** stage to the parallel canvas.
5. Add any additional stages that the job requires. Select the appropriate stages and drag them to the parallel canvas. Place the stages between the WebSphere MQ Connector stage and the Distributed Transaction stage.

   For example, you might have a job that includes a Column Import stage that splits a message into columns, and a Filter stage that routes the input data to one of several output links, depending on the value of a specific column. For this job, you must place the Column Import stage after the WebSphere MQ Connector stage and place the Filter stage after the Column Import stage, but before the Distributed Transaction stage.
6. Specify the link or links between the stages in the job:
   a. In the Palette window, select the **General** category.
   b. Select the **Link** icon and draw a link between the WebSphere MQ Connector stage and the next stage in the job. The WebSphere MQ Connector stage now has an output link.

c. If the job includes additional stages, select the **Link** icon and draw links to connect the stages in the job.

7. For input to the Distributed Transaction stage, draw one or more links that represent database work that is performed as part of the distributed transaction. The maximum number of input links that you can specify for the Distributed Transaction stage is 1000.

Configuring the WebSphere MQ Connector stage

You configure the IBM WebSphere MQ Connector stage to specify how the stage processes messages.

About this task

Procedure

To configure the WebSphere MQ Connector stage:

Procedure

1. In the parallel canvas, double-click the **WebSphere MQ Connector** stage icon to open the stage editor.
2. Select the **Stage** tab and modify the values in the **General** tab, if the defaults are not meaningful.
3. Select the **Output** tab.
4. Select the **Properties** tab, then set the properties according to the WebSphere MQ connector documentation: *Connectivity Guide for InfoSphere MQ Applications*. To allow distributed transactions, specify values for the following properties:
   a. In the **Queue manager** field, type the source queue name.
   b. In the **Wait time** field, specify the number of seconds after which the job stops if no more messages arrive. If you want the job to run indefinitely, set this value to -1.
   c. In the **Message read mode** field:
      - If you use a work queue or multiple work queues, select **Move to work queue**.
      - If you do not use a work queue, select **Keep**.
   d. If you use a work queue, in the **Work queue ->Name field**, type the work queue name. If the WebSphere MQ Connector stage is running in parallel, the connector constructs the work queue name by adding a dot (.) and the node number to the end of the specified work queue name. For example, if you specify WQ for this property, and the stage runs on two nodes, the work queue WQ.1 is used on the first node, and the work queue WQ.2 is used on the second node. To set up this behavior, you must also set the **Append node number property** of the Distributed Transaction stage to **Yes**.
   e. In the **Transaction ->Record Count** field, specify the number of messages that you want to include in each distributed transaction. This value is the number of records in a wave.
   f. In the **Transaction ->Time interval** field, specify the amount of time that elapses between end-of-wave markers. An end-of-wave marker indicates the end of a transaction. You must set this property to ensure that when the number of messages on the source queue is less than the record count, the messages are processed in a reasonable amount of time.

g. In the **Transaction ->End of wave** field, select **After** to indicate that an end-of-wave marker is added after a transaction is committed.
h. In the **Transaction ->End of wave ->End of data field**, select **Yes** to specify that the end of wave marker is inserted for the last remaining set of records.

5. In the **Columns** tab, set the properties according to the WebSphere MQ Connector documentation: *Connectivity Guide for InfoSphere MQ Applications*. You must define at least a column for the message body and a column for the MQ message ID. Other columns are optional.
   a. Set the **Data element** value for the MQ message ID column to **WSMQ.MSGID**. For this column, set the **SQLType** value to **Binary**, the **Length** value to **24**, and the **Nullable** value to **No**. The values for these properties are case-sensitive.
6. Click **OK**.

Configuring the Distributed Transaction stage

In the Distributed Transaction stage, you specify how the distributed transaction job reads messages from the queue and writes messages to target databases.

Procedure

1. On the parallel canvas, double-click the **Distributed Transaction** stage icon to open the stage editor.
2. In the stage editor, click the **Input** tab.
3. For each input link, configure the connector properties:
   a. From the list of connectors, choose a data source for the write operation.
   b. On the Properties page, set the connector properties for the write operation that you want to perform on a data source.
4. On the Columns page, specify the columns in the data source table to update. If the MQ messaging property is set to **Yes**, you must include a column to contain the value of the message ID for the source message.
5. Specify columns that do not correspond to columns in the target databases, such as sort key columns.
   a. Click the **Data element** column, and then select the column from the list.
   b. Specify names for the columns that you added.
6. If the job uses messages from failed transactions, specify values for the DTS.REJECTED and DTS.REJECTED.MESSAGE columns for one or more input links.
7. If you want to order the database work within each transaction by input link, use the **Link Ordering** tab.
8. Click **OK**.

Specifying the order of input data

If the Distributed Transaction stage has multiple input links, you can control the processing order of input data across links.

About this task

There are two ways to control the processing order of the input data to the Distributed Transaction stage:

- Specify the order in which links are processed within each transaction.

- Specify the order in which input data is processed, regardless of the link that an input record is in.

Important: When groups of messages must be processed in a specific order, ensure that the distributed transaction job runs sequentially. To set the jobs to run sequentially:

Procedure

1. In the parallel canvas, double-click the **Distributed Transaction** stage icon to open the stage editor.
2. In the stage editor, select the Distributed Transaction stage.
3. Click the **Advanced** tab.
4. Change **Execution** mode to **Sequential**.
5. Click **OK**.

Specifying the order of input data by input link

To control the order of record processing by input link, use the **Link Ordering** tab in the Distributed Transaction stage editor.

About this task

The order in which you specify the links in the **Link Ordering** tab determines the order in which the records in the links are processed for each unit of work.

Procedure

To order links:

Procedure

1. In the parallel canvas, double-click the **Distributed Transaction** stage icon to open the stage editor.
2. In the stage editor, select an input link.
3. Click the **Link Ordering** tab.
4. Click a link that you want to reorder and use the arrow buttons to move the link up or down.
5. Continue using the arrow buttons to reorder links until the links are correctly ordered.
6. Click **OK**.

Results

If multiple MQ source messages operate on the same rows in a database table, and you order the input links, the results can vary, depending on the number of records in the unit of work.

Example

For example, suppose that an MQ source queue has these six records, which describe operations that you want to perform on a table named DEPARTMENT2:

```
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........
```

```
0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
```

Now suppose that you filter those records into three Distributed Transaction stage input links, depending on whether the second field is I (INSERT), U (UPDATE), or D (DELETE). The contents of the three input links look like this:

Table 9. Link contents for Distributed Transaction stage link ordering example

Link name	Link contents
Insert_DB2	`0001,I,001,Department Name.....................,MgrNo.,ADM,Location........` `0004,I,002,Department Name.....................,MgrNo.,ADM,Location........`
Update_DB2	`0002,U,001,Department Name.....................,MgrNo.,ADM,Location........` `0005,U,002,Department Name.....................,MgrNo.,ADM,Location........`
Delete_DB2	`0003,D,001,Department Name.....................,MgrNo.,ADM,Location........` `0006,D,002,Department Name.....................,MgrNo.,ADM,Location........`

In the **Link Ordering** tab of the Distributed Transaction stage editor, you order the links like this:

1. Delete_DB2
2. Update_DB2
3. Insert_DB2

If the transaction size (**Record count** property in the WebSphere MQ Connector stage) is three, the six records are processed like this:

Table 10. Example of record processing for link ordering when transaction size is 3

Transaction number	Records in transaction, in order of processing	Result of transaction

1	`0003,D,001,...`
	`0002,U,001,...`
	`0001,I,001,...`
		A record for department 001 is inserted into table DEPARTMENT2. The delete and update records have no effect, because nothing happens when you update or delete a record that does not yet exist.
		
		There is now one record in DEPARTMENT2.

2	`0006,D,002,...`
	`0005,U,002,...`
	`0004,I,002,...`
		A record for department 002 is inserted into table DEPARTMENT2. The delete and update records have no effect, because nothing happens when you update or delete a record that does not yet exist.
		
		There are now two records in DEPARTMENT2.

However, if you change the transaction size to two, the six records are processed like this, which yields completely different results:

Table 11. Example of record processing for link ordering when transaction size is 2

Transaction number	Records in transaction, in order of processing	Result of transaction
1	`0002,U,001,...` `0001,I,001,...`	A record for department 001 is inserted into table DEPARTMENT2. There is no delete record in the first transaction. The update record has no effect, because nothing happens when you update a record that does not yet exist. There is now one record in DEPARTMENT2.

Table 11. Example of record processing for link ordering when transaction size is 2 (continued)

Transaction number	Records in transaction, in order of processing	Result of transaction
2	0003,D,001,... 0004,I,002,...	The record for department 001 is deleted from table DEPARTMENT2, and a record for department 002 is inserted into DEPARTMENT2. There is no update record in the second transaction. There is now one record in DEPARTMENT2.
3	0006,D,002,... 0005,U,002,...	The record for department 002 is deleted from table DEPARTMENT2. The update record has no effect because the record has already been deleted. There are now zero records in DEPARTMENT2.

To avoid this inconsistent behavior, ensure that the transaction size is set appropriately.

Specifying the order of input data across input links

To control the order of record processing, regardless of the links that the records are on, use the **Order records** property in the Distributed Transaction stage editor.

Before you begin

Before you can order your input data across links, a column in your MQ source messages must be usable as a sort key column.

About this task

Procedure

To order records across links:

Procedure

1. In the parallel canvas, double-click the **Distributed Transaction** stage icon to open the stage editor.
2. In the stage editor, select the Distributed Transaction stage.
3. Change the value of **Order records** to **Yes**.
4. To add additional key columns, right-click on the initial **Key column** property and select **Add property value**.
5. Click the **Available columns** button on the **Column name** property and select the column that you want to use for sorting.
6. Set the sorting order and other additional properties, as required and click **OK**.

Rejecting transactions

You can include logic in your Distributed Transaction stage jobs to roll back transactions that include failed records, and to store records that cause a failure in a reject queue. In addition, you can set records in your jobs that cause the Distributed Transaction stage to reject transactions.

Rejecting transactions with failures

To roll back transactions that include failed records and to store records that cause a failure in a reject queue, set properties in the Distributed Transaction stage.

About this task

Procedure

To set those properties, follow these steps:

Procedure

1. In the parallel canvas, double-click the **Distributed Transaction** stage icon.
2. Select the **Stage** tab.
3. In the **Properties** tab, in the **Reject failing units** field, select **Yes**.
4. In the **Reject queue** field, specify the name of a predefined local queue.
5. Click **OK**.

Generating rejections

You can set up a Distributed Transaction job to reject certain records, based on their content.

About this task

When a previous job stage determines that a record is unacceptable for insertion into a target database, you might want the Distributed Transaction stage to reject the record.

Procedure

To cause the Distributed Transaction stage to reject certain records, follow these steps.

Procedure

1. In a job stage before the Distributed Transaction stage, define the following columns.

Table 12. Required columns for rejecting records based on content

Data Element	Data type	How to use this column
DTS.REJECTED	Integer	DataStage initializes this value to 0. Set this column to 1 from any stage in the job to indicate that the whole transaction should be rejected.
DTS.REJECTED.MESSAGE	Varchar	Set this column to some error message text from within any stage in the job. If you use a reject queue in the Distributed Transaction stage, DataStage adds this message to the beginning of failed messages that it sends to the reject queue. The maximum length of this message is 488 characters.

2. In a job stage before the Distributed Transaction stage, include logic to set the DTS.REJECTED column to 1 if a record is unacceptable.
3. If you plan to use a reject queue, set the following properties:
   a. In the parallel canvas, double-click the **Distributed Transaction** stage icon.
   b. Select the **Stage** tab.
   c. In the **Reject failing units** field, select **Yes**.
   d. In the **Reject queue** field, specify the name of a predefined local queue.
   e. In the **Prepend reject** field, select **Yes** to add text from the DTS.REJECTED.MESSAGE column to the beginning of the rejected record. The following information is added to the rejected record:

- The 4 byte link number that contains the failure
- The 4 byte message status
- The 4 byte number of failed or rejected rows within the message
- The 12 byte error code that is returned by the resource (if an error occurs while writing to a target)
- The 488 byte message from the DTS.REJECTED.MESSAGE column, if specified

If you do not want this information added to the beginning of the rejected record, accept the default, **No**.

f. In the **Stop on first error** field, select **Yes** to stop the job when a record is rejected. Otherwise, accept the default, **No**, to continue processing when a record is rejected.

Configuring the Distributed Transaction stage without MQ Connector stage

You configure the Distributed Transaction stage to specify how the distributed transaction job reads messages from data sources other than the IBM WebSphere MQ Connector stage and writes multiple outputs.

Before you begin

These steps assume that you have already configured the source and that the parallel canvas for your distributed transaction job is open.

About this task

Procedure

To configure the Distributed Transaction stage:

Procedure

1. In the parallel canvas, double-click the **Distributed Transaction** stage icon to open the stage editor.
2. In the **General** tab, modify the **Description** field if the defaults are not meaningful.
3. Configure the Distributed Transaction stage properties:
   a. Select the **Stage** tab.
   b. In the **Queue manager** field, type the name of the queue manager.
   c. In the **Use MQ messaging** field, select **No**.
   d. In the **Order records** field, to preserve message order across input parallel links, select **Yes**. If you select **Yes**, define a sort key in a key column.
   e. To add additional key columns, right click on the initial **Key column** property and select **Add property value**.
   f. Click the **Available columns** button on the **Column name** property and select the column that you want to use for sorting.
   g. Set the sorting order and other additional properties, as required and click **OK**.
4. Configure the connector properties for each input link:
   a. Select the **Input** tab.
   b. In the **Properties** tab, set the connector properties for the write operation that you want to perform on a data source. For more information about

specifying DB2 connector properties, see the *IBM InfoSphere DataStage and QualityStage Connectivity Guide for IBM DB2 Databases*. To allow distributed transactions, specify the following properties:

- **Alternate conductor settings**: If the DB2 client authentication is not used, specify values for the **Alternate conductor settings**.
- **Transaction** > **Record count**: Specify 0 for this value.
- **Session** > **Array size**: If the **Order records** property is **Yes**, specify 1 for the array size.

c. In the **Columns** tab, specify the columns in the DB2 table that is being updated. Also you must specify the key column for the message ID for the source message. Specify these values for message ID column:

- Set a **Column Name** value for the MQ message ID. Set the **Data element** value for this column to **WSMQ.MSGID**, the **SQLType** value to **Binary**, the **Length** value to **24**, and the **Nullable** value to **No**.

d. In the **Columns** tab, specify any columns that do not correspond to columns in the target database.

Columns that do not correspond to columns in the target database might be sort key columns.

e. If the job uses job-generated failures, specify the DTS.REJECTED and DTS.REJECTED.MESSAGE columns for at least one input link.

5. If you want to order the database work within each transaction by input link, click the **Link Ordering** tab.
6. Click **OK**.
7. Configure the connector properties for at least one output link:
   a. Select the **Output** tab.
   b. In the **Columns** tab, click the **Load** button, browse through **->Table Definitions ->Database ->DistributedTransaction** and load the **TransactionStatus** table definition.

Note: You can specify only the Data Elements that are listed in the TransactionStatus table definition. Each column in the TransactionStatus table is optional. However, at least one column must be specified in the **Columns** tab. You can specify the same **Data Element** column for multiple column names if required.

Compiling and running distributed transaction jobs

You compile distributed transaction jobs into executable scripts that you can schedule by using the InfoSphere DataStage and QualityStage Director client and that you run on the InfoSphere Information Server engine.

About this task

Procedure

To compile and run a distributed transaction job:

Procedure

1. In the IBM InfoSphere DataStage and QualityStage Designer client, open the job that you want to compile.
2. Click the **Compile** toolbar button  .

3. If the Compilation Status area shows errors, edit the job to resolve the errors. After resolving the errors, click the **Re-compile** button.
4. When the job compiles successfully, click the **Run** toolbar button  , and specify the job run options:
   a. Enter the job parameters as required.
   b. Click the **Validate** button to verify that the job will run successfully without actually extracting, converting, or writing data.
   c. Click the **Run** button to extract, convert, or write data.
5. To view the results of validating or running a job:
   a. In the Designer client, select **Tools** > **Run Director** to open the Director client.
   b. In the Status column, verify that the job was validated or completed successfully.
   c. If the job or validation fails, select **View** > **Log** to identify any runtime problems.
6. If the job has runtime problems, fix the problems, recompile, validate (optional), and run the job until it completes successfully.

Chapter 5. Distributed transaction examples

Several example jobs are provided to help you begin writing distributed transaction jobs.

Setting up example jobs

To set up examples for the Distributed Transaction stage, create the IBM DB2 tables that are used by the sample jobs, import the sample jobs, and run a job to populate the source queue with messages.

About this task

To set up the examples for the Distributed Transaction stage:

Procedure

1. Issue the following commands to create the tables that are used by the sample jobs:

```
CREATE TABLE "DEPARTMENT2"  (
    "DEPTNO" CHAR(3) NOT NULL ,
    "DEPTNAME" CHAR(36) NOT NULL ,
    "MGRNO" CHAR(6) ,
    "ADMRDEPT" CHAR(3) NOT NULL ,
    "LOCATION" CHAR(16) ) ;

CREATE TABLE "WITHCONSTRAINT"  (
    "NAME" CHAR(10) ,
    "AGE" INTEGER ) ;
      ALTER TABLE WITHCONSTRAINT ADD CONSTRAINT AGE_CONSTRAINT CHECK (AGE < 100);

CREATE TABLE DUMMY(COL1 INTEGER);
```

2. Set up the following MQ queues for the sample job:

   SOURCEQ
   A source queue that provides input to the initial MQ Connector stage. This queue is used by all sample jobs.

   WORKQ
   A work queue that holds the output messages from the MQ Connector stage, so that the job can be restarted from an intermediate stage. This queue is used by all sample jobs.

   REJECTQ
   A queue that holds the messages that are rejected by the Distributed Transaction stage. This queue is used by the RejectedByJobLogic and RejectTransaction sample jobs.

   You can create MQ queues by running the MQSC command line utility for managing queue managers, or by using MQ Explorer. To run the MQSC command line utility, issue the following command:

   ```
   runmqsc qmgrname
   ```

   In this command, *qmgrname* is the name of the queue manager.

3. Import the `DTSJobs.dsx` file that is contained in the `DTSamples.zip` file. You can find the `DTSamples.zip` file in the `c:\IBM\InformationServer\Clients\Samples\Connectors` directory.

   The `DTSJobs.dsx` file includes a prerequisite job called GenerateMessages.

4. Run the GenerateMessages job to populate the source queue with messages.

Overview of example jobs

The sample file, `DTSJobs.dsx`, includes a variety of jobs that demonstrate how to set up distributed transactions.

The sample file includes the following jobs.

Table 13. Distributed transaction sample jobs

Job name	Purpose
DoTransaction	Demonstrates the basic setup for a Distributed Transaction job. This job has a single input link to the Distributed Transaction stage.
DoTransaction3Links_IUD	Demonstrates how to set up a Distributed Transaction job with multiple input links to the Distributed Transaction stage. Also shows how to order the processing of the links within a transaction.
DoTransaction3Links_DUI	Performs the same actions as DoTransaction3Links_IUD, but orders the links differently. Demonstrates that processing of the same input data in a different link order can affect the final content of a target database.
DoTransaction3Links_Ordered	Demonstrates how to set up a Distributed Transaction job to order data across Distributed Transaction input links.
RejectTransaction	Demonstrates how to set up a Distributed Transaction stage to roll back work when it receives a rejection record.
RejectedByJobLogic	Demonstrates how to roll back an entire transaction when a previous stage of a Distributed Transaction stage job indicates that there is a problem with the data.

Each example uses a prerequisite job to load data into the MQ source queue. The following table describes those jobs:

Table 14. Prerequisite jobs for the distributed transaction example jobs

Job name	Prerequisite for	Purpose
GenerateMsgs	DoTransaction	Populates the source queue with 10 messages.
GenerateMsgs3Links	DoTransaction3Links_IUD, DoTransaction3Links_DUI, DoTransaction3Links_Ordered	Populates the source queue with 99 messages. Each message contains a field that indicates one of three operations (U, I, or D) on the target database table. The contents of input links for the Distributed Transaction stage depend on the value of that field.
RejectTransactionLoader	RejectTransaction	Populates the source queue with nine messages. One record contains data that violates a check constraint on the target database table.
GenerateMsgsReject	RejectedByJobLogic	Populates the source queue with 101 messages. Each message contains a field that indicates one of two operations (U or I) on the target database table, or R, which means that the transaction that contains the record should be rejected by the Distributed Transaction stage. The contents of the input links for the Distributed Transaction stage depend on the value of that field.

Example: One source queue and one database link

A simple example of a distributed transaction job includes a single source queue that supplies messages for a single database update.

The DoTransaction job contains the following stages:

MQSource
An IBM WebSphere MQ Connector stage that reads messages from the source queue (SOURCEQ), moves them to a work queue (WORKQ), and sends the message data, including the MQ message ID, to the output link. The stage sends out an end-of-wave marker after every two messages.

ParseBody
A Column Import stage that parses the body of the incoming MQ message, using fixed column widths. This stage splits the message into columns, which are passed to the DistributedTransaction stage. The MQ message ID from the MQSource stage is passed through the ParseBody stage to the output link.

DistributedTransaction
A Distributed Transaction stage that writes rows to an IBM DB2 database, and deletes the corresponding messages on the work queue (WORKQ). If a failure occurs when data is written to the database, no changes are made to the database. In addition, the messages are rolled back and remain on WORKQ.

MQSource: WebSphere MQ Connector stage

To see the properties that are set on MQSource, open the DoTransaction job. In the parallel canvas, double-click the MQSource stage. In the stage editor, click the WebSphere MQ Connector stage to see the connector properties. Click the output link to see the output link properties.

The following table lists the output link property values for the WebSphere MQ connector that affect the distributed transaction.

Table 15. WebSphere MQ connector properties in the DoTransaction sample job that affect distributed transaction processing

Property name	Value in DoTransaction	Comments
Usage: Queue name	SOURCEQ	
Usage: Wait time	1	Indicates that the job stops after no more messages arrive within one second.
Usage: Message read mode	Move to work queue	Indicates that you are using a work queue.
Usage: Work queue: Name	WORKQ	
Usage: Transaction: Record count	10000	Indicates that an end-of-wave marker is inserted every 10000 records, which means that a transaction contains 10000 records.
Usage: Transaction: Time interval	0	Indicates that an end-of-wave marker is inserted as soon as the queue becomes empty.

Table 15. WebSphere MQ connector properties in the DoTransaction sample job that affect distributed transaction processing (continued)

Property name	Value in DoTransaction	Comments
Usage: Transaction: End of wave	After	This is the only valid setting for a distributed transaction.
Usage: Transaction: End of wave: End of data	Yes	

ParseBody: Column import stage

The output link of the MQSource connector, SourceMsgs, is the input link for the ParseBody stage. The ParseBody stage parses the Body column of the MQ messages, and produces multiple output columns. The stage properties specify that the Body column is parsed, and that the DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, and LOCATION columns are created.

The output link of the ParseBody stage is the DB2 connector input link for the DistributedTransaction stage. This input link represents the connection to the target database.

DistributedTransaction: Distributed Transaction stage

A Distributed Transaction stage has properties for the stage and for the DB2 connector input links. To see the properties that are set on DistributedTransaction, open the DoTransaction job. In the parallel canvas, double-click the Distributed Transaction stage. In the stage editor, click the Distributed Transaction stage to see the stage properties. Click the input link to see the DB2 connector input link properties.

The following table lists the input link property value that has special settings for the Distributed Transaction stage in the DoTransaction job.

Table 16. DB2 connector input link properties in the DistributedTransaction stage of the DoTransaction sample job

Property name	Value in DoTransaction	Comments
Connection: Database	SAMPLE	This database name must match the database that you specified in the XA resource queue manager properties when you set up dtmqmgr, which is the MQ queue manager that you are using for this job.
Connection: Alternate Conductor settings	YES	
Connection: Alternate Conductor settings	Database SAMPLE	
Connection: Alternate Conductor settings	User name DB2ADMIN	
Connection: Alternate Conductor settings	Password #Password#	

The following table lists the stage property values for the DistributedTransaction stage in the DoTransaction job.

Table 17. Stage properties in the DistributedTransaction stage of the DoTransaction sample job

Property name	Value in DoTransaction	Comments
Connection: Work queue	WORKQ	The source of MQ messages for the stage. This value is also specified in the WebSphere MQ connector output properties.
Connection: Append node number	No	The node number is not required because the configuration file does not specify multiple nodes.
Usage: Reject failing units	No	This is the only valid value.
Usage: Order records	No	Ordering of records across input links is not required because there is one link only.

The DB2 connector input link inserts data into the DEPARTMENT2 table. Therefore, the **Columns** tab of the input link specifies the DEPARTMENT2 columns. The **Columns** tab also includes the WSMQ.MSGID column. The Distributed Transaction stage retrieves the message ID of the message on the work queue from the WSMQ.MSGID column.

Operation of the DoTransaction job

When the DoTransaction job runs, the source messages are moved from the SOURCEQ queue to the WORKQ queue. If the job fails, the messages remain on the WORKQ queue. If you restart the job, the WebSphere MQ connector browses the WORKQ queue and writes the contents of the messages in the WORKQ queue to its output link. Then the WebSphere MQ connector begins to look for more messages on the SOURCEQ queue. This process prevents data loss between job runs, or if the job fails and is later restarted. To prevent data loss between queue manager restarts or system restarts, define queues as persistent.

Example: Ordering of Distributed Transaction input links

The DoTransaction3Links_IUD and DoTransaction3Links_DUI jobs demonstrate the effect of different ordering of Distributed Transaction input link processing.

Those jobs contain the following stages:

MQSource
An IBM WebSphere MQ Connector stage that reads messages from the source queue (SOURCEQ), moves them to a work queue (WORKQ), and sends the message data, including the MQ message ID, to the output link. The stage sends an end-of-wave marker after every three messages.

ParseBody
A Column Import stage that parses the body of the incoming MQ message, using fixed column widths. This stage splits the message into columns, which are passed to the Filter stage. The MQ message ID from the MQSource stage is passed through the ParseBody stage to the output link.

Filter A Filter stage that routes the input data to one of three output links, depending on the value of the second column. The MQ message ID from the MQSource stage is passed through the Filter stage to the output links.

DistributedTransaction
A Distributed Transaction stage that performs three sets of updates to an IBM DB2 database, and deletes the corresponding messages on the work queue (WORKQ). If a failure occurs when data is written to the database, no changes are made to the database. In addition, the messages are rolled back and remain on WORKQ.

The input records to the two jobs look like this:

```
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........
0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
...
```

The first field is a sequence number, which is not used in these jobs. The second field is the transaction type (I=INSERT, U=UPDATE, and D=DELETE). Records are sorted into one of three Distributed Transaction stage input links based on this field. The rest of the fields contain column data for the DEPARTMENT2 table, which is the target table for this data.

MQSource: WebSphere MQ Connector stage

To see the properties that are set on MQSource, open the job. In the parallel canvas, double-click the MQSource stage. In the stage editor, click the WebSphere MQ Connector stage to see the connector properties. Click the output link to see the output link properties. A significant property value for the DoTransaction3Links_IUD and DoTransaction3Links_DUI jobs is the **Record count** value of 3. The link ordering in the Distributed Transaction stage depends on this value.

ParseBody: Column import stage

The output link of the MQSource connector, SourceMsgs, is the input link for the ParseBody stage. The ParseBody stage parses the Body column of the MQ messages, and produces multiple output columns. The stage properties specify that the Body column is parsed, and that the SEQNUM, TRANS_TYPE, DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, and LOCATION columns are created.

The output link of the ParseBody stage is the input link for the Filter stage.

Filter: Filter stage

The Filter stage sends the input data to three output links, depending on the value of the TRANS_TYPE column. The Predicates values in the Filter stage Properties settings that determine the output links are:

```
Where Clause = TRANS_TYPE = 'I'
  Output Link = 2
Where Clause = TRANS_TYPE = 'U'
  Output Link = 1
Where Clause = TRANS_TYPE = 'D'
  Output Link = 0
```

These settings mean that records that signify DELETE operations go to output link 0, records that signify UPDATE operations go to output link 1, and records that signify INSERT operations go to output link 2.

The output links of the Filter stage are the input links for the Distributed Transaction stage. These input links represent the connections to the target database.

DistributedTransaction: Distributed Transaction stage

A Distributed Transaction stage has properties for the stage and for the DB2 connector input links. To see the properties that are set on DistributedTransaction, open the job. In the parallel canvas, double-click the Distributed Transaction stage. In the stage editor, click the Distributed Transaction stage to see the stage properties. Click an input link to see the DB2 connector input link properties.

The settings in the **Link Ordering** tab in the Distributed Transaction stage editor determine the order in which the records in the links are processed for each unit of work. The link ordering differs for each job, as shown in the following table.

Table 18. Link ordering for Distributed Transaction sample jobs with more than one database link

Job name	Order of processing of Distributed Transaction input links
DoTransaction3Links_IUD	1. Insert_DB2 2. Update_DB2 3. Delete_DB2
DoTransaction3Links_DUI	1. Delete_DB2 2. Update_DB2 3. Insert_DB2

Operation of the DoTransaction3Links_IUD and DoTransaction3Links_DUI jobs

When the DoTransaction3Links_IUD and DoTransaction3Links_DUI jobs run, data reaches the Distributed Transaction stage in one of three input links, depending on the value of the TRANS_TYPE field in the Filter stage. The order of processing of the messages depends on the specified link order.

In job DoTransaction3Links_DUI, the first six messages are processed in this order:

```
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........

0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
```

The first two operations in each transaction have no effect because they perform operations on rows that do not yet exist. The final result of processing these six records is that two records are inserted into the DEPARTMENT2 table.

However, in job DoTransaction3Links_IUD, the first six messages are processed in this order:

```
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........

0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
```

With this ordering, a record is inserted, updated, and then deleted. The final result of processing these six records is that there are no records in the DEPARTMENT2 table.

Example: Ordering of data across Distributed Transaction input links

The DoTransaction3Links_Ordered job demonstrates the result of ordering data across Distributed Transaction input links.

The DoTransaction3Links_Ordered job contains the following stages:

MQSource
An IBM WebSphere MQ Connector stage that reads messages from the source queue (SOURCEQ), moves them to a work queue (WORKQ), and sends the message data, including the MQ message ID, to the output link. The stage sends out an end-of-wave marker after every three messages.

ParseBody
A Column Import stage that parses the body of the incoming MQ message, using fixed column widths. This stage splits the message into columns, which are passed to the Filter stage. The MQ message ID from the MQSource stage is passed through the ParseBody stage to the output link.

Filter A Filter stage that routes the input data to one of three output links, depending on the value of the second column. The MQ message ID from the MQSource stage is passed through the Filter stage to the output link.

DistributedTransaction
A Distributed Transaction stage that performs three sets of updates to an IBM DB2 database, and deletes the corresponding messages on the work queue (WORKQ). If a failure occurs when data is written to the database, no changes are made to the database. In addition, the messages are rolled back and remain on WORKQ.

The input records to the job look like this example:

```
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........
0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
...
```

The first field is a sequence number, which is not used to order the input to the Distributed Transaction stage. The second field is the transaction type (I=INSERT, U=UPDATE, and D=DELETE). Records are inserted into one of three Distributed Transaction stage input links based on this field. The rest of the fields contain column data for the DEPARTMENT2 table, which is the target table for this data.

MQSource: WebSphere MQ Connector stage

To see the properties that are set on MQSource, open the job. In the parallel canvas, double-click the MQSource stage. In the stage editor, click the WebSphere MQ Connector stage to see the connector properties. Click the output link to see the output link properties.

ParseBody: Column import stage

The output link of the MQSource connector, SourceMsgs, is the input link for the ParseBody stage. The ParseBody stage parses the Body column of the MQ messages, and produces multiple output columns. The stage properties specify that the Body column is parsed, and that the SEQNUM, TRANS_TYPE, DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, and LOCATION columns are created.

The output link of the ParseBody stage is the input link for the Filter stage.

Filter: Filter stage

The Filter stage sends the input data to three output links, depending on the value of the TRANS_TYPE column. The Predicates values in the Filter stage Properties settings that determine the output links are:

```
Where Clause = TRANS_TYPE = 'I'
  Output Link = 2
Where Clause = TRANS_TYPE = 'U'
  Output Link = 1
Where Clause = TRANS_TYPE = 'D'
  Output Link = 0
```

These settings mean that records that signify DELETE operations go to output link 0, records that signify UPDATE operations go to output link 1, and records that signify INSERT operations go to output link 2.

In the **Mapping** tab for the Filter output, SEQNUM, the sequence number input column, is mapped to the DTS_SEQNUM column, because that column is used for sorting in the Distributed Transaction stage. Columns in input links to the Distributed Transaction stage that are not columns of the target table must begin with DTS_.

The output links of the Filter stage are the input links for the Distributed Transaction stage. These inputs links represent the connections to the target database.

DistributedTransaction: Distributed Transaction stage

A Distributed Transaction stage has properties for the stage and for the DB2 connector input links. To see the properties that are set on DistributedTransaction, open the job. In the parallel canvas, double-click the Distributed Transaction stage. In the stage editor, click the Distributed Transaction stage to see the stage properties. Click the input link to see the DB2 connector input link properties.

The **Order records** property in the Distributed Transaction stage properties is set to **Yes** to enable ordering across links. To specify data ordering, DTS_SEQNUM column is defined in the initial Key Column property. In each input link, **Perform sort** is selected, and the DTS_SEQNUM column is defined as the ordering key.

Operation of the DoTransaction3Links_Ordered job

When the DoTransaction3Links_Ordered job runs, data reaches the Distributed Transaction stage in one of three input links, depending on the value of the TRANS_TYPE field in the Filter stage. The order of processing of the records depends on the DTS_SEQNUM column value, which is the ordering key. The DTS_SEQNUM column is the first column in the input data.

In job DoTransaction3Links_Ordered, the first six records are processed in this order:

```
0001,I,001,Department Name.....................,MgrNo.,ADM,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
0003,D,001,Department Name.....................,MgrNo.,ADM,Location........

0004,I,002,Department Name.....................,MgrNo.,ADM,Location........
0005,U,002,Department Name.....................,MgrNo.,ADM,Location........
0006,D,002,Department Name.....................,MgrNo.,ADM,Location........
```

Each record causes a row to be inserted, updated, and then deleted. The final result of processing these six records is that there are no records in the DEPARTMENT2 table.

Example: Rejecting a transaction with a failing record

The RejectTransaction job demonstrates how to roll back an entire transaction when a failure occurs during processing.

The RejectTransaction job contains the following stages:

MQSource
An IBM WebSphere MQ Connector stage that reads messages from the source queue (SOURCEQ), moves them to a work queue (WORKQ), and sends the message data, including the MQ message ID, to the output link. The stage sends out an end-of-wave marker after every three messages.

ParseBody
A Column Import stage that parses the body of the incoming MQ message, using fixed column widths. This stage splits the message into columns, which are passed to the DistributedTransaction stage. The MQ message ID from the MQSource stage is passed through the ParseBody stage to the output link.

DistributedTransaction
A Distributed Transaction stage that inserts a set of rows into an IBM DB2 database, and deletes the corresponding messages on the work queue (WORKQ). If a failure occurs while processing a transaction, all messages in that transaction are moved to a reject queue, and processing continues with the next unit of work.

The input records to the job look like the following example:

```
Name1,030
Name2,030
Name3,030
Name4,030
Name5,200
Name6,030
Name7,030
Name8,030
Name9,030
```

The two fields correspond to the NAME and AGE columns in the target table, WITHCONSTRAINT. The check constraint causes any input rows with an Age value of 100 or more to be rejected.

MQSource: WebSphere MQ Connector stage

To see the properties that are set on MQSource, open the job. In the parallel canvas, double-click the MQSource stage. In the stage editor, click the WebSphere MQ Connector stage to see the connector properties. Click the output link to see the output link properties.

ParseBody: Column import stage

The output link of the MQSource connector, SourceMsgs, is the input link for the ParseBody stage. The ParseBody stage parses the Body column of the MQ messages, and produces multiple output columns. The stage properties specify that the Body column is parsed, and that the Name and Age columns are created.

The output link of the ParseBody stage is the input link for the DistributedTransaction stage.

DistributedTransaction: Distributed Transaction stage

A Distributed Transaction stage has properties for the stage and for the DB2 connector input links. To see the properties that are set on the Distributed Transaction stage, open the job. In the parallel canvas, double-click the Distributed Transaction stage. In the stage editor, click the Distributed Transaction stage to see the stage properties. Click an input link to see the DB2 connector input link properties.

The **Reject failing units** property in the Distributed Transaction stage properties is set to **Yes** to cause all work in a transaction to be rolled back if an error occurs during insertion of any row into the target table. The **Reject queue** value is set to REJECTQ, to indicate where the records in the failing transaction should be placed.

Operation of the RejectTransaction job

When the RejectTransaction job runs, each transaction consists of the following three records:

```
Name1,030
Name2,030
Name3,030

Name4,030
Name5,200
Name6,030

Name7,030
Name8,030
Name9,030
```

In the first transaction, all three rows are inserted into the WITHCONSTRAINT table. However, in the second transaction, the second row fails the check constraint on the table, because the Age value is not less than 100. Therefore, the entire second transaction fails, and the messages for that transaction are placed in the REJECTQ queue. Processing continues with the third transaction, and the rows are inserted successfully.

Example: Rejecting a transaction based on user-defined criteria

The RejectedByJobLogic job demonstrates how to roll back an entire transaction when a previous stage of the Distributed Transaction stage job indicates a problem with the data.

Sometimes data that fits the criteria for insertion in a database table must be rejected for some other reason. For example, there might be no table check constraints on a table column, but previous steps in the job determine that certain input values are invalid. You can design a Distributed Transaction stage job so that when a previous stage determines that a record is bad, that stage sets a flag to identify the bad record. When the Distributed Transaction stage processes the bad record, it rolls back the transaction that contains that record.

The RejectedByJobLogic is a simple example of such a job. RejectedByJobLogic contains the following stages:

MQSource
An IBM WebSphere MQ Connector stage that reads messages from the source queue (SOURCEQ), moves them to a work queue (WORKQ), and sends the message data, including the MQ message ID, to the output link. The stage sends out an end-of-wave marker after every 52 messages.

ParseBody
A Column Import stage that parses the body of the incoming MQ message, using fixed column widths. This stage splits the message into columns, which are passed to the Filter stage. The MQ message ID from the MQSource stage is passed through the ParseBody stage to the output link.

Filter A Filter stage that routes the input data to one of three output links, depending on the value of the second column (TRANS_TYPE). Records that have a TRANS_TYPE value of U or I go directly to the DistributedTransaction stage. A subset of data in each record with a TRANS_TYPE value of R goes to the SetReject stage. The MQ message ID from the MQSource stage is passed through the Filter stage to each output link.

SetReject
A Transformer stage that processes rejected records. This stage sets the special column named IsRejected, which tells the DistributedTransaction stage to reject the transaction that contains the record. Output from this stage goes to the DistributedTransaction stage.

DistributedTransaction
A Distributed Transaction stage that performs update or insert operations on an IBM DB2 database, or rolls back transactions, depending on the content of the incoming messages. If the DistributedTransaction stage receives a record with the IsRejected column set to 1, the Distributed Transaction stage rolls back the transaction that contains the record, and moves the input messages to a reject queue (REJECTQ). If a transaction contains no reject records, and all updates are successful, the Distributed Transaction stage deletes the corresponding messages from the work queue (WORKQ).

The input records to the job look like the following example. Only one input record is a reject record.

```
0001,I,001,Department Name.....................,MgrNo.,III,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
...
```

```
0082,I,028,Department Name.....................,MgrNo.,III,Location........
0083,U,028,Department Name.....................,MgrNo.,ADM,Location........
0085,I,029,Department Name.....................,MgrNo.,III,Location........
----,R,---,-----------------------------------,------,---,----------------
0086,U,029,Department Name.....................,MgrNo.,ADM,Location........
...
0149,U,050,Department Name.....................,MgrNo.,ADM,Location........
```

The second field is the transaction type (I=INSERT, U=UPDATE, and R=REJECT). For records with a transaction type of U or I, the rest of the fields contain column data for the DEPARTMENT2 table, which is the target table for this data. For records with a transaction type of R, the rest of the fields are meaningless.

MQSource: WebSphere MQ Connector stage

To see the properties that are set on MQSource, open the job. In the parallel canvas, double-click the MQSource stage. In the stage editor, click the WebSphere MQ Connector stage to see the connector properties. Click the output link to see the output link properties.

ParseBody: Column import stage

The output link of the MQSource connector, SourceMsgs, is the input link for the ParseBody stage. The ParseBody stage parses the Body column of the MQ messages, and produces multiple output columns. The stage properties specify that the Body column is parsed, and that the SEQNUM, TRANS_TYPE, DEPTNO, DEPTNAME, MGRNO, ADMRDEPT, and LOCATION columns are created.

The output link of the ParseBody stage is the input link for the Filter stage.

Filter: Filter stage

The Filter stage sends the input data to three output links, depending on the value of the TRANS_TYPE column. The Predicates values in the Filter stage Properties settings that determine the output links are:

```
Where Clause = TRANS_TYPE = 'I'
  Output Link = 0
Where Clause = TRANS_TYPE = 'U'
  Output Link = 1
Where Clause = TRANS_TYPE = 'R'
  Output Link = 2
```

These settings mean that records that signify INSERT operations go to output link 0 (Insert_DB2), records that signify UPDATE operations go to output link 1 (Update_DB2), and records that trigger rejected transactions go to output link 2 (RejectedMsg).

Output links 0 and 1 of the Filter stage are the input link for the Distributed Transaction stage. These input links represent the connections to the target database.

Output link 2 of the Filter stage is the input link to the SetReject stage. Only the first field of the message and the message ID are passed to the SetReject stage.

SetReject: Transformer stage

The SetReject stage creates a four-column output record for each reject record that it receives. To see those columns, in the parallel canvas, double-click the SetReject stage. The following table describes these columns:

Table 19. Columns in the SetReject stage

Column name	Value	Usage
IsRejected	1	Tells the DistributedTransaction stage to reject the transaction that contains the record.
RejectMessage	"This message was rejected by the filter stage."	Tells the DistributedTransaction stage to add this text to the beginning of every record that caused an error.
MsgID	The message ID from the source message	This column is required.
COL1	0	This column contains a user-created table named DUMMY, which is used only to satisfy the requirements of a DB2 connector. The output link for the SetReject stage is a DB2 connector that is the input to the DistributedTransaction stage. The DB2 connector properties must include an SQL statement. Therefore, the output link must pass a table column to the DistributedTransaction stage.

The output records are sent to the DistributedTransaction stage through the RejectTX link.

DistributedTransaction: Distributed Transaction stage

A Distributed Transaction stage has properties for the stage and for the DB2 connector input links. To see the properties that are set on the Distributed Transaction stage, open the job. In the parallel canvas, double-click the Distributed Transaction stage. In the stage editor, click the Distributed Transaction stage to see the stage properties. Click the input link to see the DB2 connector input link properties.

In the DistributedTransaction stage, the **Reject failing units** property value is set to **No**. If this option is selected, transactions that contain a reject record (a record in which the IsRejected column value is 1) are rolled back. The source messages are left on the work queue, and nothing is written to the target DB2 database.

The RejectTX input link, which is the source of reject records, must be defined as a DB2 connector. Therefore, the **Write mode** property is set to Update, and **SQL** > **Update statement** contains the following SQL statement, which results in no action.

```
UPDATE DUMMY SET COL1=0 WHERE 0=1
```

The RejectTX input link creates a four-column record. To see those columns, in the parallel canvas, double-click the RejectTX input link. The following table describes these columns:

Table 20. Columns in the RejectTX input link

Column name	SQL type	Length	Data Element
IsRejected	Integer		DTS.REJECTED
RejectMessage	VarChar	488	DTS.REJECTED.MESSAGE
MsgID	Binary	24	WSMQ.MSGID
COL1	Integer		

Operation of the RejectedByJobLogic job

When the RejectedByJobLogic job runs, the first transaction consists of 54 records, and the second transaction consists of the remaining 47 records:

```
0001,I,001,Department Name.....................,MgrNo.,III,Location........
0002,U,001,Department Name.....................,MgrNo.,ADM,Location........
...
0080,U,027,Department Name.....................,MgrNo.,ADM,Location........

0082,I,028,Department Name.....................,MgrNo.,III,Location........
0083,U,028,Department Name.....................,MgrNo.,ADM,Location........
0085,I,029,Department Name.....................,MgrNo.,III,Location........
----,R,---,-----------------------------------,------,---,----------------
0086,U,029,Department Name.....................,MgrNo.,ADM,Location........
...
0149,U,050,Department Name.....................,MgrNo.,ADM,Location........
```

The first transaction contains no records with a value of R in the second field, so the transaction completes successfully. The second transaction contains a record with a value of R in the second field, which the SetReject stage translates into a record with an IsRejected value of 1. When the DistributedTransaction stage reads that record, it rejects the transaction and rolls back all updates to the target table that are in that transaction.

Chapter 6. Improving the performance of Distributed Transaction stage jobs

Distributed transactions tend to be slower than local transactions. You can improve the performance of Distributed Transaction stage jobs by adjusting the transaction size or the array size, and by using Distributed Transaction stage jobs only when necessary.

The following suggestions can help you maximize the performance of your Distributed Transaction stage jobs.

- Adjust the transaction size.

  You adjust the transaction size in the IBM WebSphere MQ Connector stage of the job. Transaction size is one of the biggest factors in Distributed Transaction stage job performance. If you use a very small transaction size (for example, 10 messages), performance can be degraded by the overhead of starting and committing many small transactions. If you use a very large transaction size (for example, 10000 messages), factors such as the amount of paging to disk can also degrade performance. For better performance, start with transaction sizes of about 100 to 1000 messages, if this approach meets your job requirements.
- Adjust the array size in the IBM DB2 connector input links.

  Although the array size is a much smaller factor in the performance of Distributed Transaction stage jobs as the transaction size, it can affect performance. Use an array size of 1 only when it is necessary, such as when the same data is being updated and referenced. You must set the array size to 1 when ordering across input links is specified.
- Use Distributed Transaction stage jobs only when you really need distributed transactions.

  Because local transactions can run twice as fast as distributed transactions, design your jobs to use Distributed Transaction stages only when you must coordinate work among several resource managers. If the same work can be performed independently by several resource managers without compromising data integrity, use local transactions.

Chapter 7. Environment variables: Distributed Transaction stage

The Distributed Transaction stage uses these environment variables.

CC_DTS_COMMIT_ON_EOF

Set this environment variable to specify whether Distributed Transaction stage commits on the end of data.

When end of wave is not inserted before end of data, Distributed Transaction stage prints the error message: `The Distributed Transaction Stage detected end of data before end of wave. Rolling back the transaction` to the job log and rolls back the transaction. By default, the Distributed Transaction stage commits only on end of wave, and does not commit on the end of data.

When the value of this variable is 1, Distributed Transaction stage partially commits on the end of data.

CC_IGNORE_TIME_LENGTH_AND_SCALE

Set this environment variable to change the behavior of the connector on the parallel canvas.

When this environment variable is set to 1, the connector running with the parallel engine ignores the specified length and scale for the timestamp column. For example, when the value of this environment variable is not set and if the length of the timestamp column is 26 and the scale is 6, the connector on the parallel canvas considers that the timestamp has a microsecond resolution. When the value of this environment variable is set to 1, the connector on the parallel canvas does not consider that the timestamp has a microsecond resolution unless the microseconds extended property is set even if the length of the timestamp column is 26 and the scale is 6.

CC_MSG_LEVEL

Set this environment variable to specify the minimum severity of the messages that the connector reports in the log file.

At the default value of 3, informational messages and messages of a higher severity are reported to the log file.

The following list contains the valid values:

- 1 - Trace
- 2 - Debug
- 3 - Informational
- 4 - Warning
- 5 - Error
- 6 - Fatal

CC_TRUNCATE_STRING_WITH_NULL

Set this environment variable to truncate string data that includes the string 0x00.

When the value of this environment variable is set and when the input data contains a null character, the input data is truncated with 0x00 and the rest of the string is dropped. This environment variable applies to fields of Char, VarChar, and LongVarChar InfoSphere DataStage types.

CC_TRUNCATE_NSTRING_WITH_NULL

Set this environment variable to truncate string data that includes the string 0x00.

When the value of this environment variable is set and when the input data contains a null character, the input data is truncated with 0x00 and the rest of the string is dropped.

CC_USE_EXTERNAL_SCHEMA_ON_MISMATCH

Set this environment variable to use an external schema rather than a design schema when the schemas do not match.

This schema is used for schema reconciliation. When the value of this environment variable is set, the behavior remains the same and is not changed from the old version.

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. Reading command-line syntax

This documentation uses special characters to define the command-line syntax.

The following special characters define the command-line syntax:

[ ] Identifies an optional argument. Arguments that are not enclosed in brackets are required.

... Indicates that you can specify multiple values for the previous argument.

| Indicates mutually exclusive information. You can use the argument to the left of the separator or the argument to the right of the separator. You cannot use both arguments in a single use of the command.

{ } Delimits a set of mutually exclusive arguments when one of the arguments is required. If the arguments are optional, they are enclosed in brackets ([ ]).

Note:

- The maximum number of characters in an argument is 256.
- Enclose argument values that have embedded spaces with either single or double quotation marks.

For example:

wsetsrc*[-S server] [-l label] [-n name] source*

The *source* argument is the only required argument for the **wsetsrc** command. The brackets around the other arguments indicate that these arguments are optional.

wlsac *[-l | -f format] [key...] profile*

In this example, the -l and -f format arguments are mutually exclusive and optional. The *profile* argument is required. The *key* argument is optional. The ellipsis (...) that follows the *key* argument indicates that you can specify multiple key names.

wrb -import *{rule_pack | rule_set}...*

In this example, the rule_pack and rule_set arguments are mutually exclusive, but one of the arguments must be specified. Also, the ellipsis marks (...) indicate that you can specify multiple rule packs or rule sets.

Appendix C. How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

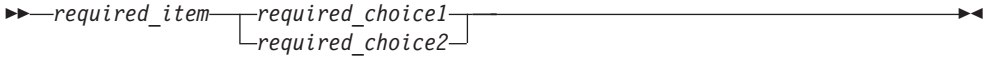
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

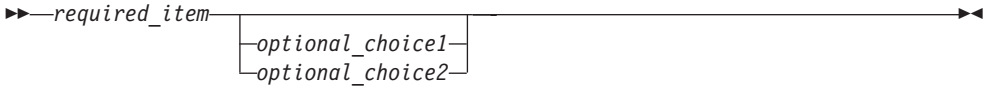
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack.

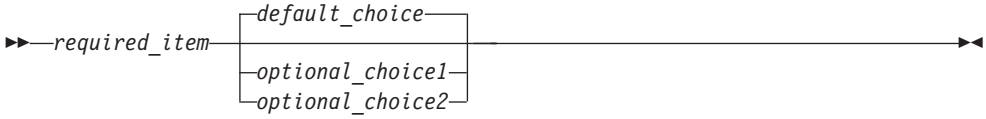
  If you must choose one of the items, one item of the stack appears on the main path.



  If choosing one of the items is optional, the entire stack appears below the main path.

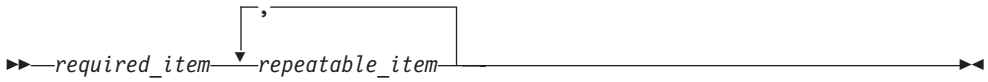


  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



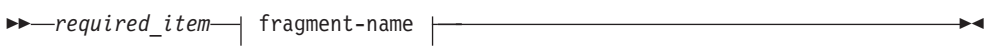
- An arrow returning to the left, above the main line, indicates an item that can be repeated.

If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



Fragment-name:

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown.
- Variables appear in all lowercase italic letters (for example, **column-name**). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

Appendix D. 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 21. 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 E. 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 F. 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 22. 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 22. 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

R

S

T

V

W

X

Printed in USA

SC19-4255-00