IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for Stored Procedures

SC19-4268-00

IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for Stored Procedures

SC19-4268-00

Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 43.

Contents

Chapter 1. Introduction

Stored procedures extend the range of InfoSphere® DataStage®. The following section provides an introduction to the Stored Procedure (STP) stage. The second half of the chapter covers installation instructions and configuration information.

Introduction

Use the Stored Procedures stage as:

- A source, returning a rowset
- A target, passing a row to a stored procedure to write
- A transform, invoking logic within the database

The Stored Procedures stage supports input and output parameters or arguments so that you can provide a value and receive a value in return. You can process the returned value after the stored procedure is run. The Stored Procedures stage provides status codes indicating whether the stored procedure completed successfully and, if not, allowing you to handle errors.

Currently The Stored Procedures stage supports Oracle, DB2®, Sybase, Teradata, and SQL Server. The Stored Procedure stage is designed so that it can be configured at runtime to execute stored procedures in a number of different supported relational database management systems using native interfaces.

Note: You must use the custom graphical user interface to enter the data for the Stored Procedures stage. IBM® does not support the use of the grid style editor. You should not use it because it may corrupt the Stored Procedure stage parameter job data. The grid editor will be disabled in a future release of DataStage Designer.

Configuration Requirements

Oracle

Oracle requires the following configuration:

- Oracle client software on the InfoSphere DataStage server machine. For most up-to-date information about supported versions of the Oracle database, contact IBM software support.
- Configuration of SQL*Net using a configuration program, for example, SQL*Net Easy Configuration, to set up and add database aliases.

DB2

DB2 uses the DRDA® protocol (TCP/IP) if the DB2 data resides on an AS/400® system.

Requirements for Environment Variables

the Stored Procedures stage requires the setting of environment variables in order to work correctly in a UNIX environment. Table 1 identifies the specific environment variables that are required.

Table 1. Required Environment Variables for UNIX

For...	Set the following environment variable on the server machine...
Oracle	ORACLE_HOME TWO_TASK ORACLE_SID LD_LIBRARY_PATH[1]
DB2	DB2INSTANCE INSTHOME LD_LIBRARY_PATH
Sybase	SYBASE SYBASE_OCS LD_LIBRARY_PATH[2], [3]
Teradata	LD_LIBRARY_PATH PATH

[1]The name of one particular environment variable, referred to as LD_LIBRARY_PATH above, differs depending on the platform. To determine the correct name to use for your environment:

- [1]If the platform is AIX®, use LIBPATH.
- [1]If the platform is HP_UX, use SHLIB_PATH.
- [1]If the platform is LINUX or Solaris, use LD_LIBRARY_PATH.

[2]To run successfully on Linux, Sybase OC requires the language environment variable LANG = en.

[3]In Sybase, for the LD_LIBRARY_PATH environment variable, the InfoSphere DataStage library entries must be referenced before the Sybase Open Client library entries at run time.

The Stored Procedures Stage and the Parallel Canvas

Some connectivity stages can run on the Parallel Canvas. The default for *all* stages is Sequential. "In Parallel" mean you can set it to run in parallel, but this is *not* the default. Those that are available have additional instructions for mapping string data when NLS is turned on (see **"Mapping of String Data"**). DRS is available on the Parallel Canvas in UNIX. You can use it as a source or a target or for Processing. As a source, it runs sequentially;as a target, it runs in parallel.

Mapping of String Data

The purpose of the NONE map on the Server canvas is to turn off mapping of string data in any stage in which the map is set, i.e., to pass the data through verbatim. This feature is handled differently on the Parallel Canvas. When you define string data (char, varchar, etc.), there is an additional field in the stage's Columns grid called Extended. This can be set to blank or Unicode. If this option is set to blank, no mapping occurs (i.e. "NONE"); the map specified on the NLS tab is ignored. If this option is set to Unicode, the NLS map is applied. In order to read or write Japanese data, for example, set Extended to Unicode. When the job compiler detects this combination (char, varchar, etc. and Unicode), it generates the appropriate run-time code.

Chapter 2. Stored Procedure stage

The Stored Procedure stage extends the range of InfoSphere DataStage by providing a facility to execute stored procedures. A stored procedure is a block of procedural constructs and embedded SQL statements that is stored in a database and that can be called by name.

This chapter describes the following subjects:

- **"Functionality"**
- **"About the STP Stage"**
- **"Using the STP Stage"**
- **"Defining Input Data"**
- **"Defining Output Data"**
- **"Links"**

Functionality

The Stored Procedures stage has the following functionality and benefits:

- Executes stored procedures and functions.
- Supports specific relational databases through native interfaces.
- Uses standard link terminology for an active stage.
- Supports the mapping of input and output columns to input and output parameters in a procedure.
- Supports the identification of additional columns to contain return values and status information.
- Supports pass-through of column information if the stage is used midstream in a job.
- Accepts job parameters to identify certain properties at run time.
- Supports the writing of locator information.
- Support for NLS (National Language Support).

The following functionality is not supported:

- Multiple procedures in a single stage.
- Execution of stored procedures at the column level. If this is a need, use a transformer stage to direct each column to a separate Stored Procedures stage.
- Execution of SQL statements.
- Reference links.

About the Stored Procedures Stage

The following sections describe the types of procedures that are supported by the Stored Procedures stage and the role of the parameters grid.

Types of Procedures

The Stored Procedures stage supports three types of procedures:

- Transform procedures
- Source procedures

- Target procedures

Source and target procedures use the column metadata as parameters to pass into the procedure. You do not have to enter the parameter information, and if you enter it, the stage ignores it. Source procedures use the column metadata as output parameters (to read from the database). Target procedures use the column metadata as input parameters (to write to the database).

Note: Source and target procedures cannot have both input and output links. You must use a transform procedure.

A source procedure has the ability to execute a stored procedure that contains an SQL SELECT statement. It can:

- Be called once for the job
- Be used with an output link
- Return output parameters
- Return more than one row

A target procedure has the ability to execute a stored procedure that contains an SQL INSERT statement. It can:

- Be called once for the job
- Be called once for each row (default)
- Be used with an input link
- Pass input parameters

If you wish to mix and match parameters to column metadata, use a transform procedure and specify the Stored Procedure Parameters in the parameters grid.

The transform procedure executes various logic decisions at the database. It can:

- Be called once for the job
- Be called once for each row (default)
- Pass input parameters using the parameters grid
- Return output parameters using the parameters grid
- Pass-through columns if used midstream (input and output links)
- Be used with an input link only
- Be used with an output link only
- Be used with an input link and an output link
- Return more than one row

All procedures can:

- Return an error code (defined by the user or from the database) that can be mapped to the data flow
- Return a message (defined by the user or from the database) that can be mapped to the data flow
- Allow user-defined fatal and warning-code recovery decisions
- Define transaction isolation levels
- Allow a commit size to be specified when executing a procedure that inserts rows
- Allow a user-defined procedure execution statement (not normally used)
- Allow more than one row to be returned when reading

Note: The Stored Procedures stage does not execute any SQL statements. Use it exclusively for the execution of stored procedures. Use the native stages for the execution of any SQL statements.

Teradata

Using the Stored Procedure stage, you can invoke Teradata stored procedures, macros, and functions from InfoSphere DataStage jobs. You cannot create Teradata stored procedures, macros, or functions from InfoSphere DataStage. Such objects should be created with Teradata's own user interfaces such as BTEQ or Teradata SQL Assistant (formerly known as Queryman prior to Teradata Tools and Utilities 7.0).

SQL Server

A stored procedure is a group of Transact-SQL statements compiled into a single execution plan and stored under a name and processed as a unit.

The Stored Procedure stage supports calling Microsoft SQL Server stored procedures (internal and external) and user-defined functions that return a single value.

Note: User-defined functions that return a table are better suited to be invoked from the ODBC stage, since the generated SELECT statement must query a table.

Scalar user-defined functions are supported. A global cursor and output parameter using cursor varying are not supported as methods of returning data.

Microsoft SQL Server stored procedures support input and output parameters.

Parameters

STP uses a parameters grid to map input and output columns to the input and output parameters of the stored procedure. The parameters grid is located on the **Parameters** tab of the Stage page. When the required information is provided, the stage parses these values and uses the information to build the stored procedure call and bind the internal variables. For source and target procedures, parameter specifications are not required and are ignored because the column metadata is used to map the parameter information.

For the transform procedure, the parameter specifications are used to map parameters to column metadata. The parameters grid must be completed with the proper information to call the stored procedure.

Rules for Specifying Parameter Values

- Literal values and InfoSphere DataStage job parameters are supported only for transform procedures.
- The job must contain an input link for the **Parameter type** to be defined as **Input/Output** or **Input** if the parameter value is **?**. InfoSphere DataStage needs an active row from which to read the values.
- The job must contain an output link for **Parameter type** to be defined as **Input/Output**, **Output**, or **Function**. (**CursorOutput** is an additional option in DB2.) InfoSphere DataStage needs to create an active row to output the values.

Examples of Parameter Values

Example 1 is an example of a parameters grid specification that sends data from an input link to the stored procedure:

Table 2. Example 1

Parameter name	Maps to Column	Parameter marker/literal	Parameter Type
EMPNOP	EMPNO	?	Input
JOBP	JOB	?	Input/Output
DEPTNOP	DEPTNO	99	Input
ENAMEP	ENAME	#ename#	Input

In Example 1, 99 is a literal, and #ename# is a job parameter.

Example 2 is an example of a parameters grid specification that sends and returns data from a stored procedure and initiates the execution of a function. It works this way because one and only one **Parameter Type** is Function:

Table 3. Example 2

Parameter name	Maps to Column	Parameter marker/literal	Parameter Type
FUNCSTATP	FUNCSTAT	?	Function
DNAMEP	DNAME	?	Output
LOCP	LOC	#city#	Input

Using the Stored Procedures stage

When you use the client GUI to edit a Stored Procedures stage, the STP Stage dialog box opens.

This dialog box can have up to three pages (depending on whether there are inputs to and outputs from the stage):

- **Stage.** This page displays the name of the stage you are editing. The **General** tab defines the database vendor, the database source, and logon information to connect to the database. It also establishes the transaction ISO level. For details see **"About the Stage Page"** .

  The **NLS** tab defines a character set map to be used with the stage. (The **NLS tab** is available only if you have installed NLS.) For details, see **"Defining Character Set Mapping"** .
- **Input.** This page is displayed only if you have an input link to this stage. It specifies the associated column definitions for each data input link. This page also specifies instructions for the stored procedure.
- **Output.** This page is displayed only if you have an output link to this stage. It specifies the associated column definitions for each data output link. This page also specifies error return code information and instructions for the stored procedure.

About the Stage Page

Double click the **STP** icon, right click the **STP** icon and select **Properties**, or choose **Properties** from the **Edit** menu. The STP Stage dialog box opens.

The Stage page consists of the **General**, **Syntax**, **Parameters**, **Error Codes**, **NLS** (optional) and **Advanced** tabs (Parallel jobs only). The **General** tab on the Stage page appears by default.

The General Tab

Use the **General** tab to set connection parameters.

The tab has the following fields:

- **Database vendor.** The type of relational database. With this release, the options are **Oracle**, **DB2**, **Sybase**, **Teradata,** and **MSSQL Server**. **Database vendor** is required.
- **Database source.** The name of the database alias.

Oracle

- For Oracle, enter the name you created using the Oracle Configuration Assistant.

DB2

- For DB2, enter the connection name, the data source name, or the database name.

SQL Server

- For SQL Server, enter an ODBC DSN name that describes driver and server information.
  - **User name.** The user name to use to connect to the database. This user must have sufficient privileges to access the specified database and source and target tables. This field is required. There is no default.
  - **Password.** The password that is associated with the specified user name. This field is required. There is no default.
  - **Database name.** The database on the specified server.

DB2

- This field is not available for DB2.

  Note: If you select **DB2** as the **Database vendor**, and click **OK**, when you return to the Stage page, **Database name** is not active.

SQL Server

- This field is not available for SQL Server.
- **Transaction ISO.** The transaction isolation level (ISO level). The ISO level provides the necessary consistency and concurrency control between transactions in the job and other transactions for optimal performance. Use **Transaction ISO** to set the transaction isolation level for input and output links by choosing a value in the list provided. ISO Level is database specific, and each database might not support all the generic choices presented. For more information on using these levels, see your database documentation. Use one of the following transaction isolation levels:
  - **None. The default.**
  - **Read-only.** Takes no write locks. This is the most efficient ISO level allowing for optimal performance.
  - **Read Uncommitted.** Takes exclusive locks on modified data. These locks are held until a commit or rollback is executed. However, other transactions can still read but not modify the uncommitted changes. No other locks are taken.

- **Read Committed.** Takes exclusive locks on modified data and sharable locks on all other data. This is the default. Exclusive locks are held until a commit or rollback is executed. Uncommitted changes are not readable by other transactions. Shared locks are released immediately after the data has been processed, allowing other transactions to modify it.
- **Repeatable Read.** Identical to serializable except that phantom rows might be seen.
- **Serializable.** Takes exclusive locks on modified data and sharable locks on all data. All locks are held until a commit or rollback is executed, preventing other transactions from modifying any data that has been referenced during the transaction.

Oracle

Select from among **None**, **Read-only**, **Read Committed**, and **Serializable** for Oracle.

DB2

Select from among **None**, **Read Uncommitted**, **Read Committed**, **Repeatable Read**, and **Serializable** for DB2.

Sybase

Select from among **None**, **Read Uncommitted**, **Read Committed**, and **Serializable** for Sybase.

SQL Server

Select from among **None**, **Read Uncommitted**, **Read Committed**, **Repeatable Read**, and **Serializable** for SQL Server.

Teradata

Transaction ISO is inactive when the **Database vendor** is **Teradata**. The Teradata RDBMS supports a LOCKING modifier on some SQL statements to allow control over transaction isolation levels, but CALL and EXECUTE statements do not support the LOCKING modifier. To use a particular locking method, specify the LOCKING modifier in the SQL statements in the body of the macro or procedure.

- **Transaction mode.** The mode of connection to the Teradata server. This field is active only when the selected **Database vendor** is **Teradata** or a job parameter.
  - If **Database vendor** is **Teradata**, the options for **Transaction mode** are **ANSI**, the default, and **Teradata**.

    This field controls whether the Stored Procedure stage connects to the Teradata server in ANSI or Teradata transaction mode. A connection in ANSI transaction mode cannot call stored procedures that were compiled in Teradata transaction mode, and a connection in Teradata transaction mode cannot call stored procedures that were compiled in ANSI transaction mode.
  - If **Database vendor** is a job parameter, the options for **Transaction mode** are **ANSI** and **Native**. Native transaction mode is equivalent to Teradata transaction mode if the database vendor at runtime is Teradata. The Transaction mode field has no effect for other vendors.

Note: If an error occurs in Teradata transaction mode, Teradata automatically rolls back the current transaction. In ANSI transaction mode, an error will not affect the current transaction.

- **Client character set.** The Teradata client character set to use when connecting to the Teradata server. This field is active only when **Database vendor** is **Teradata** or a job parameter. The default value is **Default**.
- **Description.** An optional description of the Stored Procedures stage.

The Syntax Tab

Use the **Syntax** tab to identify the stored procedure to be executed.

The following properties are included on the **Syntax** tab:

- **Procedure name.** The name of the stored procedure to execute on the server. You must provide the specific name of the procedure you wish to execute. You can also click the **...** button at the right of **Procedure name** to browse the Repository to select the stored procedure. In order to do this, the stored procedure definition must be imported into the Repository via the IBM InfoSphere DataStage and QualityStage® Designer client by using the **Import Table Definitions Stored Procedure Definitions**.

Teradata

The Stored Procedure Definitions import facility uses ODBC, so you must first set up an ODBC data source name (DSN). We recommend that you use the latest Teradata ODBC driver, version 3.04, which comes with the Teradata Tools and Utilities 8.0 client.

- **Procedure type.** The type of procedure to execute. The options are:
  - **Source.** Source procedures emulate an SQL SELECT statement and use the column metadata as output parameters to read from the database. They have only an output link.
  - **Target.** Target procedures emulate an SQL INSERT statement and use the column metadata as input parameters to write to the database. They have only an input link.
  - **Transform.** Transform procedures use stored procedure properties to execute various logic decisions at the database. They can be used with an input link only, with an output link only, or with both input and output links. Specify the input and output parameters on the parameter grid.

    The default is **Transform**.
- **Database procedure type.** A refinement on the type of procedure to execute. **Database procedure type** affects the generated procedure call syntax. It is active only if **Database vendor** is **MSSQL Server**, **Teradata**, or a job parameter.

Teradata

The options for Teradata are:

- **Stored procedure**
- **Macro**
- **Scalar user-defined function**
- **Table user-defined function**

SQL Server

The options for SQL Server are:

- **Stored procedure**
- **Scalar user-defined function**

The default is **Stored procedure**.

- **Procedure call syntax.** The syntax used to execute the procedure call statement. The syntax can be modified by clearing **Generate procedure call**.
- **Generate procedure call.** The option to have the stage generate the procedure call statement by using the job metadata. Procedure name and either parameter information or column information are used to generate the call syntax. If selected, the stage generates the statement. If cleared, the stage does not generate the statement. To edit procedure syntax, clear **Generate procedure call**. The default is **Generate procedure call** selected.

  Note: When you enter user-defined call statements (**Generate procedure call** is cleared) to execute procedures, you must follow the specific database syntax for the stage to successfully execute the procedure.

The Parameters Tab

The **Parameters** tab identifies and describes the parameters used when the **Procedure type** is **Transform**.

The Stored Procedures stage allows you to map input and output columns to the input and output parameters of the procedure via parameter specifications. The stage uses the information provided on the **Parameters** tab to build the stored procedure call and bind the internal variables. The **Parameters** tab is ignored if Procedure type is set to **Source** or **Target** because the column metadata is used to map the parameter information.

The following properties are included on the **Parameters** tab:

- **Parameter name.** The name of the parameter.
- **Maps to Column.** The name of the column to which the parameter maps.
- **Parameter marker/literal.** A literal value, the IBM InfoSphere DataStage job parameter, or a parameter marker (**?**). The default is **?**.
- **Parameter type.** The type of parameter. The options are:
  - **Input.** The stage sends data from the link or a literal value to the stored procedure. This is the default. The stage must contain an input link. Select **Input** to satisfy WHILE or WHERE clauses for the selected operation and to insert values in the table.
  - **Output.** The stage returns data or a result set from the stored procedure. The stage must contain an output link. Select **Output** to satisfy `SELECT @param=10`. The stored procedure returns single rows or specific output parameters.
  - **CursorOutput.** The stage returns data or a result set from the stored procedure. The stage must contain an output link. **CursorOutput** is available only for DB2 and SQL Server. Select **CursorOutput** to satisfy `SELECT one from TABLE_NAME`. The stored procedure returns a record or a result set.
  - **Input/Output.** The stage sends data from the link and returns data or a result set. The database must support the use of parameters that can send and return information. The stage must contain an input and an output link.
  - **Function.** Some databases support stored functions. **Function** triggers the syntax generated to execute a function instead of a stored procedure. The

mapping of the parameter set to **F** contains the return value of the stored function. Only one parameter can be defined as a function type. The stage must contain an output link.

Oracle

Valid values for **Parameter Type** for Oracle are **Input**, **Output**, **Input/Output**, and **Function**.

DB2

Valid values for **Parameter Type** for DB2 are **Input**, **Output**, **CursorOutput**, and **Input/Output**.

Note: Although Stored Procedure Stage does not support stored functions with DB2, you can execute stored functions on one of two ways:

- Call a stored procedure that executes a user-defined function
- Code a call to a stored function within a SQL statement

Sybase

Valid values for **Parameter Type** for Sybase are **Input** and **Output**. Sybase does not support stored functions.

Teradata

Valid values for **Parameter Type** for Teradata are **Input**, **Output**, **Input/Output**, and **Function**.

SQL Server

Valid values for **Parameter Type** for SQL Server are **Input**, **Output**, **CursorOutput**, **Input/Output**, and **Function**.

- **Load....** Click the **Load...** button below the grid to browse the Repository to select the procedure parameters. In order to do this, the procedure parameters must be imported into the Repository.

Teradata

Normally, the **Load** button loads only from the **Parameters** tab of the **Table Definitions** record. But when **Database vendor** is **Teradata**, the **Load** button also loads parameters from the **Columns** tab of the **Table Definitions** record. These parameters have their **Parameter Type** set to

- **Input** if **Key** is selected on the **Columns** tab of the **Table Definition**
- **Output** if **Key** is not selected on the **Columns** tab of the **Table Definition**

This is necessary since the Stored Procedure Definitions import facility imports the output columns of a macro into the **Columns** tab, and the Teradata API plug-in imports all parameters into the **Columns** tab.

The Error Codes Tab

The **Error Codes** tab identifies user-defined errors.

The following properties appear on the **Error Codes** tab:

- **User defined errors.** The codes identifying errors. The lists should contain both database-specific errors expressed as the integer portion only as well as any user-defined errors that can be returned by the procedure. There are two categories of errors:
  - **Fatal errors.** Specify errors that should be treated as fatal. The stage checks the list of fatal errors first. If it finds one of the codes identified as fatal, the job aborts.
  - **Warnings.** Specify errors that should be treated as warnings. The stage checks the list of warnings only after it checks the list of fatal errors. If the stage finds any of the codes identified as a warning, the stage writes a log entry and processing continues.

    Error handling in the Stored Procedures stage is similar to that of the ODBC stage. The Stored Procedures stage provides the properties **Fatal errors** and **Warnings** in which you can enter a list of values, separated by a space, to handle return values within the stored procedure.

    The stage checks these lists of codes and processes the error codes as you have requested.
  - If the code is in the fatal list, the job aborts.
  - If the code is in the warning list, a log entry is written and processing continues.

    The stage checks the fatal list first. If you specify the same code in both lists, the job aborts. The lists should contain database-specific errors reduced to just the integer part of the error. Also you can provide user-defined error codes that the procedure might return. If you leave the lists empty (the default), the stage uses the database's standard of what is and is not fatal and operates in that manner.

    When writing data to the database, if the data is truncated while taking it from the IBM InfoSphere DataStage, you have the option to continue processing. Enter the special warning code -99 in the warning list to trigger the stage to pass truncated data to the database. The stage writes a warning entry to the log, and the job does not abort.

    Note: Take care treating fatal errors as warnings. Such action could cause the stage to take unknown code paths and fail. "Unique key violation" is an example of an error you could treat as a warning, skipping the row and continuing processing. But if you treat "Can not connect to Database" as a warning, the Stage cannot recover from it, and processing fails.

Teradata

- **Load....** Click the **Load...** button to browse the Repository for particular fatal or warning codes to be imported. In order to do this, the stored procedure codes must be imported into the Repository.

The Advanced Tab

The **Advanced** tab is used only in parallel jobs. The **Advanced** tab allows you to set some particular features about how the stage behaves. Generally you can ignore this tab, and let the stage take the default values. It is intended for advanced users to finely tune operations.

Defining Character Set Mapping

You can define a character set map for a stage. Do this from the **NLS** tab on the Stage page. The **NLS** tab is available only if you have installed NLS.

Specify information using the following fields:

- **Map name to use with stage.** Defines the default character set map for the project or the job. You can change the map by selecting a map name from the list.
- **Show all maps.** Lists all the maps that are shipped with InfoSphere DataStage.
- **Loaded maps only.** Lists only the maps that are currently loaded.
- **Use Job Parameter....** Specifies parameter values for the job. Use the format #*Param*#, where *Param* is the name of the job parameter. The string #*Param*# is replaced by the job parameter when the job is run.

Defining Input Data

When you write data to a table in a database, the Stored Procedures stage has an input link. The properties of this link and the column definitions of the data are defined on the Input page in the STP stage dialog box of the GUI.

About the Input Page

The Input page has an **Input name** field and **General** and**Columns** tabs:

- **Input name.** The name of the input link. The Stored Procedures stage can have only one input link.

The General Tab

Use the **General** tab to set connection parameters.

The tab has the following fields:

- **Database vendor.** The type of relational database. With this release, the options are **Oracle**, **DB2**, **Sybase**, **Teradata,** and **MSSQL Server**. **Database vendor** is required.
- **Database source.** The name of the database alias.

Oracle

- For Oracle, enter the name you created using the Oracle Configuration Assistant.

DB2

- For DB2, enter the connection name, the data source name, or the database name.

SQL Server

- For SQL Server, enter an ODBC DSN name that describes driver and server information.
  - **User name.** The user name to use to connect to the database. This user must have sufficient privileges to access the specified database and source and target tables. This field is required. There is no default.
  - **Password.** The password that is associated with the specified user name. This field is required. There is no default.
  - **Database name.** The database on the specified server.

DB2

- This field is not available for DB2.

  Note: If you select **DB2** as the **Database vendor**, and click **OK**, when you return to the Stage page, **Database name** is not active.

SQL Server

- This field is not available for SQL Server.
- **Transaction ISO.** The transaction isolation level (ISO level). The ISO level provides the necessary consistency and concurrency control between transactions in the job and other transactions for optimal performance. Use **Transaction ISO** to set the transaction isolation level for input and output links by choosing a value in the list provided. ISO Level is database specific, and each database might not support all the generic choices presented. For more information on using these levels, see your database documentation. Use one of the following transaction isolation levels:
  - **None. The default.**
  - **Read-only.** Takes no write locks. This is the most efficient ISO level allowing for optimal performance.
  - **Read Uncommitted.** Takes exclusive locks on modified data. These locks are held until a commit or rollback is executed. However, other transactions can still read but not modify the uncommitted changes. No other locks are taken.
  - **Read Committed.** Takes exclusive locks on modified data and sharable locks on all other data. This is the default. Exclusive locks are held until a commit or rollback is executed. Uncommitted changes are not readable by other transactions. Shared locks are released immediately after the data has been processed, allowing other transactions to modify it.
  - **Repeatable Read.** Identical to serializable except that phantom rows might be seen.
  - **Serializable.** Takes exclusive locks on modified data and sharable locks on all data. All locks are held until a commit or rollback is executed, preventing other transactions from modifying any data that has been referenced during the transaction.

Oracle

Select from among **None**, **Read-only**, **Read Committed**, and **Serializable** for Oracle.

DB2

Select from among **None**, **Read Uncommitted**, **Read Committed**, **Repeatable Read**, and **Serializable** for DB2.

Sybase

Select from among **None**, **Read Uncommitted**, **Read Committed**, and **Serializable** for Sybase.

SQL Server

Select from among **None**, **Read Uncommitted**, **Read Committed**, **Repeatable Read**, and **Serializable** for SQL Server.

Teradata

Transaction ISO is inactive when the **Database vendor** is **Teradata**. The Teradata RDBMS supports a LOCKING modifier on some SQL statements to allow control over transaction isolation levels, but CALL and EXECUTE statements do not support the LOCKING modifier. To use a particular locking method, specify the LOCKING modifier in the SQL statements in the body of the macro or procedure.

- **Transaction mode.** The mode of connection to the Teradata server. This field is active only when the selected **Database vendor** is **Teradata** or a job parameter.
  - If **Database vendor** is **Teradata**, the options for **Transaction mode** are **ANSI**, the default, and **Teradata**.

    This field controls whether the Stored Procedure stage connects to the Teradata server in ANSI or Teradata transaction mode. A connection in ANSI transaction mode cannot call stored procedures that were compiled in Teradata transaction mode, and a connection in Teradata transaction mode cannot call stored procedures that were compiled in ANSI transaction mode.
  - If **Database vendor** is a job parameter, the options for **Transaction mode** are **ANSI** and **Native**. Native transaction mode is equivalent to Teradata transaction mode if the database vendor at runtime is Teradata. The Transaction mode field has no effect for other vendors.

    Note: If an error occurs in Teradata transaction mode, Teradata automatically rolls back the current transaction. In ANSI transaction mode, an error will not affect the current transaction.
- **Client character set.** The Teradata client character set to use when connecting to the Teradata server. This field is active only when **Database vendor** is **Teradata** or a job parameter. The default value is **Default**.
- **Description.** An optional description of the Stored Procedures stage.

Columns Tab

This tab contains the column definitions for the data written to the table. The column definitions appear in the same order as in the Columns grid. The **Columns** tab behaves the same way as the **Columns** tab in the ODBC stage.

Teradata

On the **Columns** tab of the Input page, the **Load** button loads only from the **Columns** tab of the Table Definitions record. If you need to load the input parameters of a macro, stored procedure, or user-defined function, you can import the object with the Teradata API stage. The Teradata API stage imports the input parameters into the **Columns** tab. The Stored Procedure Definitions import facility only imports input parameters into the **Parameters** tab.

Defining Output Data

When you use the Stored Procedures stage to execute a procedure at the beginning of a job path, the stage executes the specified stored procedure once and sends a single or multiple rows down the output link. Output links specify the data you are extracting from a database.

About the Output Page

The Output page has one field and the **General** and **Columns** tabs.

- **Output name.** The name of the output link.

General Tab

This tab contains the following parameters:

- **Procedure status to link.** Disposition of the return information from the execution of the stored procedure. If **Procedure status to link** is selected (the default), the return code from the stored procedure and the error message are sent down the output link as column data. The first two columns on the output link hold the output of the stored procedure return code and message information. These columns have the description field set to PROC.RTN.CODE and PROC.RTN.MESS and must be the first two columns defined on the output link. The stage creates or removes these columns on the output link depending on whether **Procedure status to link** is selected or cleared.

SQL Server

SQL Server operates slightly differently. If **Procedure status to link** is selected (the default), the return code from the stored procedure is sent down the output link as column data. The first column on the output link holds the output of the stored procedure return code. This column has the description field set to PROC.RTN.CODE and must be the first two columns defined on the output link. The stage creates or removes these columns on the output link depending on whether **Procedure status to link** is selected or cleared

The column metadata is created as shown in the following table.

Table 4. Column Metadata with SQL Server

Name	Data type	Description
ProcCode	Integer	PROC.RTN.CODE
ProcMess	Char(128)	PROC.RTN.MESS

Sybase

Sybase can not return status information on a per row basis when the procedure returns a result set. Since the status is not returned until all rows have been fetched, the return code and the error message cannot be mapped into the data flow.

- **Activity Count to Link.** When selected, the number of rows affected by the macro or procedure is sent down the output link.

Teradata

Activity Count to Link is active only for Teradata or for a job parameter.

- **Procedure EOD return code.** The code the procedure returns when it reaches end of data. If you select **Procedure returns multiple rows**, you must provide the code the procedure returns when it finishes fetching rows. The stage sends multiple rows down the output link until the result set from the procedure is exhausted and the procedure returns this code to the Stored Procedures stage. Providing the code triggers an execution loop in the stage that continues outputting rows until the called procedure tells the Stored Procedures stage to stop.

Oracle

Procedure EOD return code is active only for Oracle. .

- **Procedure returns multiple rows.** Indicator that the procedure returns more than one row. Select **Procedure returns multiple rows** if the procedure returns multiple rows. Clear **Procedure returns multiple rows** if the procedure returns only one row. The default is **Procedure returns multiple rows** cleared.

Oracle

For more information regarding Oracle, see Managing Cursors .

Sybase

Sybase uses output parameters to return column data of the result set until all rows have been fetched.

Teradata

Procedure returns multiple rows behaves as it does for DB2. By default, **Procedure returns multiple rows** is cleared.

SQL Server

By default, **Procedure returns multiple rows** is not selected. If not selected, only the first row is returned from stored procedures that return multiple rows. This is similar to doing a singleton lookup. If selected, each call to a stored procedure can send multiple rows down the output link. This is equivalent to doing a cursor lookup. If a stored procedure returns multiple result sets, the number of columns in each result set must match the number of output parameters on the **Parameters** tab, and the columns must be data type compatible with the output parameters.

- **Description.** An optional description of the output link.

Using **Procedure EOD return code**, writing procedures that fetch from cursors, and returning multiple rows are database specific.

Columns Tab

The column tab page behaves the same way as the **Columns** tab in the ODBC stage, and it specifies which columns are aggregated.

Teradata

On the **Columns** tab of the Output page, the **ProcCount** column appears if **Activity count to link** is selected.

The **Load** button loads only from the **Columns** tab of the **Table Definitions** record. If you need to load the output parameters of a stored procedure or user-defined function, you can import the object with the Teradata API stage. The Teradata API stage imports all parameters into the **Columns** tab. The Stored Procedure Definitions import facility imports parameters only into the **Parameters** tab.

Links

The Stored Procedure stage can have only one input link and/or one output link. Reference links are treated as input links and are not supported by STP; this is an active stage convention.

However, it is possible to emulate a reference link by using the stage midstream and defining input parameters as keys to the WHERE clause in the stored procedure and defining output parameters to hold the resulting row selected. See **"Emulating Reference Links"** .

Midstream Use

If the Stored Procedures stage is placed midstream (the job contains an input and output link), column data for output parameters arrives from the execution of the procedure. If **Forward row data** is selected, the data for other columns is passed thru to the output link.

Column Metadata

The Stored Procedures stage creates column metadata definitions on the output link by copying the input columns to the output columns when **Forward row data** is selected. If **Parameter type** is **Output**, **Input/Output**, or **Function**, the associated column metadata must exist on the output link. By default, any procedure that defines parameters that return data need storage to return the data to, which explains the need for these parameter types to exist on the output link.

Column Data

The following rules apply:

- If **Parameter type** for the column is set to **Output**, **Input/Output**, or **Function**, the column data comes from the execution of the procedure.
- When **Forward row data** is selected:
  - If **Parameter type** for the column is set to **Input**, the column data is passed from the input link to the output link.
  - If the column does not appear in the parameters grid (and is therefore not used as a parameter), the column data is passed from the input link to the output link.

Emulating Reference Links

Although the traditional reference link scenario does not apply to active stages, STP can handle reference lookups using a different paradigm. The following example illustrates the use of reference lookups with the Stored Procedures stage.

In this example using Oracle, the objective is to join the employee table to the department table and project the employee name along with the name of the department in which this employee works. In a traditional job, Sequential_File_0 feeds employee records to the Transfer_1 stage, and the Oracle_OCI_3 stage selects the department name on a keyed read of the DEPT table using the department number from the employee record. Transformer_1 pushes the employee name/department name pair to the Sequential_File_2 stage. The following figure illustrates this.

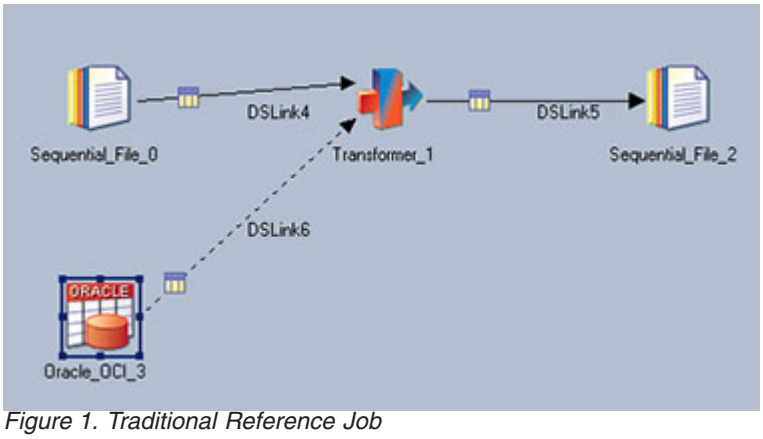


Figure 1. Traditional Reference Job

To develop this job using the Stored Procedures stage, Sequential_File_0 feeds employee records to the Stored Procedures stage. The Stored Procedures stage executes the procedure DSSP for each row. The Stored Procedures stage passes the department number of the employee record as a parameter and returns the department name as an output parameter from the keyed lookup done by the procedure. By selecting Forward row data, all of the input column data is copied to the output column data (including employee name). The row consisting of the employee record columns and the department name from the department record is sent to Transformer_10. The transformer then projects the employee name/department name pair to the Sequential_File_2 stage. The transformer stage is used to limit column selection on the output link, because all the columns are copied from the input link to the output link when **Forward row data** is selected.

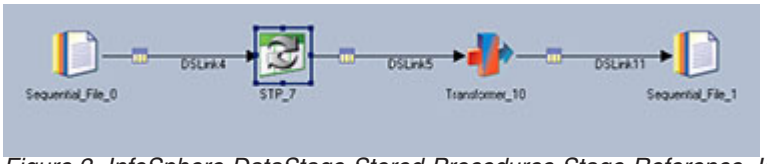


Figure 2. InfoSphere DataStage Stored Procedures Stage Reference Job

For this example, the content of the **Columns** tab of the Input page of the Stored Procedures stage is shown in the following table.

Table 5. Example of **Columns** *Tab of the Input Page*

Column name	Key	SQL Type	Length	Scale	Nullable	Display
EMPNO	Yes	Decimal	4		No	6
ENAME	No	VarChar	10		Yes	10
JOB	No	VarChar	9		Yes	4
MGR	No	Decimal	4		Yes	6
SAL	No	Decimal	7		Yes	9
COMM	No	Decimal	7		Yes	9
DEPTNO	No	Decimal	2		No	4

The content of the **Columns** tab of the Output page of the Stored Procedures stage is shown in the following table.

Table 6. Example of **Columns** *Tab of the Output Page*

Column name	Derivation	Key	SQL Type	Length	Scale	Nullable	Display
ProcCode		No	Integer	10		No	10
ProcMess		No	VarChar	128		No	32
EMPNO		Yes	Decimal	4		No	6
ENAME		No	VarChar	10		Yes	10
JOB		No	VarChar	9		Yes	9
MGR		No	Decimal	4		Yes	6
SAL		No	Decimal	7		Yes	9
COMM		No	Decimal	7		Yes	9
DEPTNP		No	Decimal	2		No	4
DNAME		No	VarChar	14		Yes	14

The content of the columns contained in the output link of the Transformer stage is shown on the following table.

Table 7. Example of the content of the Columns in the Output Link of the Transformer Stage

Column name	Key	SQL Type	Length	Scale	Nullable	Display
ENAME		VarChar	10		Yes	10
DNAME		VarChar	14		Yes	14

To accomplish the objective, set the Stored Procedure stage properties as follows:

- **General** tab of the Input page:
  - **Forward row data** is selected
  - **Execute procedure for each row** is selected
- **Syntax** tab of the Stage page:
  - **Procedure type** is set to **Transform**
- **Parameters** tab of the Stage page

Table 8. Properties on the **Parameters** *Tab*

Column Name	Parameter Value	Parameter Type
DEPTNO	?	Input
DNAME	?	Output

The stored procedure DSSP21 is made up of the following code:

```
PROCEDURE DSSP21 (DEPTNO_ARG IN DEPT.DEPTNO%TYPE,
                  DNAME_ARG OUT DEPT.DNAME%TYPE)
AS BEGIN
   SELECT DNAME INTO DNAME_ARG
      FROM DEPT WHERE DEPTNO = DEPTMNO_ARG;
END;
```

Chapter 3. Additional database specifics

Certain topics, such as using Procedure EOD return code, writing procedures that fetch from cursors, and returning multiple rows, are database specific. This appendix contains specific information about Oracle, DB2, Sybase, Teradata, and SQL Server.

Oracle

For specific information about writing PL/SQL procedures, refer to your database documentation. The following features are specific to Oracle.

Packages

In addition to executing stored procedures, the Stored Procedures stage supports and executes stored functions and procedures in packages. Set **Procedure name** on the **Syntax** tab of the Stage page to package.procedurename to execute a procedure in a package.

Handling PL/SQL Errors

Using the error handling capability of the Stored Procedures stage, you can check for Oracle-specific database errors, define your own error codes, or do both. The PL/SQL call RAISE_APPLICATION_ERROR() returns user-defined or Oracle database status to the Stored Procedures stage. The following PL/SQL fragment catches various error states:

```
EXCEPTION
WHEN NO_DATA_FOUND THEN
   RAISE_APPLICATION_ERROR (-20001,'No data found - just warning');
WHEN ZERO_DIVIDE THEN
   /* return Oracle database error (ORA-01476 and message, treat as warning*/
WHEN OTHERS THEN
   RAISE_APPICATION_ERROR (-20002,"SOMETHING BAD HAPPENED - ABORT');
```

Provide the following values for **User defined errors** on the **Error Codes** tab of the **Stage** page:

- Fatal errors: 20002
- Warnings: 20001 01476

You can enter multiple error codes in each category. Separate them with a space.

In this example, NO DATA FOUND and ZERO DIVIDE are treated as warnings, the job continues to run, and the stage processes subsequent rows. All other errors are fatal and cause the job to abort.

If you select **Procedure status to link** on the **General** tab of the Output page, RAISE_APPLICATION_ERROR() does a rollback on pending rows. The error codes and messages from RAISE_APPLICATION_ERROR() are passed down the output link as the first two columns.

Managing Cursors

By managing cursors properly, you can select more than one row or return result sets.

If the stored procedure returns multiple rows, you must code the PL/SQL procedure in a specific way to interface with the Stored Procedures stage. You must explicitly declare the cursor external to the procedure being executed by the Stored Procedures stage to process the rows.

Managing the cursor consists of two parts:

1. The designer must construct the procedure to initialize the cursor with the OPEN statement and fetch one row only. When the procedure decides to stop fetching rows, it must raise the "Procedure EOD code" error to notify the Stored Procedures stage to stop outputting rows and release the cursor with the CLOSE statement. The stored procedure should not have a FETCH loop in the PL/SQL code because InfoSphere DataStage needs to process one row at a time.

   The scope of the cursor must be external to the procedure because the Stored Procedures stage executes the procedure until the procedure returns the "Procedure EOD code." If the cursor is not external, each execution of the procedure returns the first row of the result set, because each time the procedure is invoked, the cursor is closed and reopened. To declare the cursor external, place it in a package. This changes the scope of the cursor so that it is not limited to a particular PL/SQL block. When the packaged cursor is opened, it remains open on top of the procedure executions by the Stored Procedures stage.
2. You must supply **Procedure EOD return code** on the output link. This tells the Stored Procedures stage to loop on the execution of the procedure until the procedure returns the code you entered using RAISE_APPLICATION_ERROR(). This code is returned when the cursor in the procedure returns %NotFound after a fetch call.

In this example, the stored procedure scans the full table and send back 20001 when all the rows are exhausted. The EOD 20001 can be sent back at any time if the user desires to read less than the whole table but return, for example, the first ten rows, giving the user full control over processing. The following PL/SQL fragment manages the cursor appropriately:

```
PACKAGE BODY DSSP3 AS
   CURSOR GET_COL IS SELECT DEPTNO,DNAME,LOC FROM DEPT;
   PROCEDURE DSSP3 (DEPTNO_ARG OUT DEPT.DEPTNO%TYPE,
               DNAME-ARG OUT DEPT.DNAME%TYPE,
               LOC-ARG OUT DELP.LOC%TYPE)
   IS BEGIN
      IF NOT GET_COL%ISOPEN THEN
         OPEN GET_COL;
      END IF;
      FETCH GET_COL INTO DEPTNO_ARG,DNAME_ARG,LOC_ARG;
      IF GET_COL%NOTFOUND THEN
         CLOSE GET-COL
         RAISE_APPLICATION_ERROR(-20001,'EOD FOUND');
END;
```

In the Stored Procedures stage, **Procedure EOD return code** is set to 20001.

Select Into

About this task

You can select a single row using STP. Clear **Procedure returns multiple rows**. Write the SELECT INTO statement with a WHERE clause on a KEY to select one or zero (NO_DATA_FOUND) rows and to assign the selected values to output

variables. BULK COLLECT and array bind variables are not supported; therefore the SELECT INTO must return only one row.

```
PROCEDURE DSSP5 (DEPTNO_ARG OUT DEPT.DEPTNO%TYPE,
            DNAME_ARG OUT DEPT.DNAME%TYPE,
            LOC_ARC OUT DEPT.LOC5TYPE)
AS BEGIN
   SELECT DEPTNO, DNAME, LOC INTO
      DEPTNO_ARG, DNAME_ARG, LOC_ARG
      FROM DEPT WHERE DEPTNO=10;
   EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RAISE_APPLICATION_ERROR (-20001, `NO DATA FOUND');
   WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR (-200002, `SOMETHING BAD HAPPENED');
END;
```

User-Defined Call Statement

About this task

If you want to enter your own call statement for the procedure, do not select **Generate procedure call**. Enter an anonymous PL/SQL block that contains the invocation syntax of the procedure you want to execute.

DB2

The following features are specific to DB2.

Handling SQL Errors

Using the error handling capability of the Stored Procedures stage, you can check for DB2-specific database errors, define your own error codes, or do both. Declare condition variables and a handler for each condition. Rather than communicating directly with programmers, SQL returns error codes to the application program in the form of SQLCODE and SQLSTATE. The Stored Procedure Stage uses RESIGNAL control statement to return a user-defined SQLSTATE and a user-defined error message to the calling process. The following example defines all key elements of the user-defined error handling:

```
DECLARE rec_cnt int;
DECLARE SQLSTATE CHAR(5);
DECLARE EXIT HANDLER FOR SQLSTATE '75000'
RESIGNAL SQLSTATE '75000' SET MESSAGE_TEXT = 'Invalid department number.';
SELECT count (*) INTO rec_cnt FROM DB2ADMIN.STPTEST WHERE STPTEST.ACHAR='XYZ';
IF rec_cnt = 0
THEN SIGNAL SQLSTATE '75000';
END IF;
```

Provide the following values on the **Error Codes** tab of the Stage page:

- Fatal errors: 2002
- Warnings: 2001 1479

where 2001 means NO DATA FOUND and 1476 means ZERO DIVIDE.

In this example, NO DATA FOUND and ZERO DIVIDE are treated as warnings, the job continues to run, and the stage processes subsequent rows. All other errors are fatal and cause the job to abort.

If you select **Procedure status to link** on the **General** tab of the Output page, RESIGNAL control statement does a rollback on pending rows. The error codes and messages from RESIGNAL control statement are passed down the output link as the first two columns.

Managing Cursors

By managing cursors properly, you can select more than one row or return result sets.

Select **CursorOutput** as the **Parameter type**. The Stored Procedures stage does not require any specification for the **Procedure EOD return code**.

Code the SQL procedure to interface with the InfoSphere DataStage Stored Procedure stage as follows.

- The Stored Procedures stage does not require the declaration of the cursor to be external to the procedure being executed to process the rows. Unlike Oracle, the rows are not fetched one at a time, and therefore all the rows will be output.
- The stored procedure should not have a FETCH loop in the SQL code because the stored procedure code does the fetching internally and transfers each row to the Data Stage client one by one.
- Do not close the cursor within the stored procedure; STP is responsible for this. The Stored Procedures stage fetches all records till SQL_NO_DAT_FOUND and then close the cursor.

By default, the Stored Procedures stage returns only one row from the client unless you select **Procedure returns multiple rows**. Selecting this check box will not generate any message as in case of oracle because EOD is not required for DB2 stored procedure plug-in.

In this example, the stored procedure obtains all the rows with selected columns from the DEPT table into cursor cursor1. The DataStage Stored Procedures stage code fetches all the rows, and no FETCH loop is required. The following SQL fragment manages the cursor appropriately:

```
CREATE PROCEDURE DB2ADMIN.SPPR2 ()
P1: BEGIN
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT DNO, DNAME, LOC FROM DEPT;
    OPEN cursor1;
END P1
```

Select Into

About this task

You can select a single row or specific data using STP. Write the SELECT INTO statement with a WHERE clause on a KEY to select one or zero (NO_DATA_FOUND) rows and to assign the selected values to output variables.

This example uses a SELECT INTO statement to return three different columns from a table into output parameters.

```
CREATE PROCEDURE SAMPLE.SP11LED (OUT DEPTNO INT,
                                 OUT DEPTNAME VARCHAR (30),
                                 OUT LOC VARCHAR (30))
P1: BEGIN
DECLARE EXIT HANDLER FOR NOT FOUND
SELECT DNO, DNAME, LOC INTO
```

```
        DEPTNO, DEPTNAME, LOC
FROM DEPT
WHERE DNO = 101;
END P1
```

User-Defined Call Statement

About this task

If you want to enter your own call statement for the procedure, do not select **Generate procedure call**. Enter an anonymous PL/SQL block that contains the invocation syntax of the procedure you want to execute.

Sybase

The following features are specific to Sybase.

Transaction Size

If **Transaction size** is equal to or greater than 1, the Stored Procedure stage sets ANSI-compatible transaction mode, also known as chained mode, so that a transaction is implicitly started

- After the connection is established and
- Before the first such statement after a transaction has been committed or rolled back.

In chained mode the Stored Procedure stage controls commit and rollback, but the procedure must be set to execute in any mode. For more information on how to set this mode on a procedure, refer to Sybase documentation

Teradata

The Stored Procedure stage supports calling Teradata macros, stored procedures (internal and external), scalar functions, and table functions. Aggregate functions are better suited to be invoked from the Teradata API stage, since the generated SELECT statement must query a table.

Although the Teradata API stage was not designed to invoke Teradata macros, stored procedures, and user-defined functions, it is possible to invoke such objects with the "User-defined SQL" option. However, there are several restrictions when invoking these objects from the Teradata API stage.

- The Teradata API stage can only import a macro's input parameters into the Columns tab.
- The Teradata API stage can return only one row from a macro invoked on the stage's reference link.
- The Teradata API stage can return only one result set from a macro invoked on the stage's output link.
- If a macro is invoked from the Teradata API stage's reference or output link, the first statement in the macro must be a SELECT statement.
- The Teradata API stage only connects in ANSI transaction mode, so it can only call procedures that were compiled in ANSI transaction mode. Attempting to invoke a stored procedure that was compiled in Teradata transaction mode will return an error that says, "Invalid session mode for procedure execution".
- The Teradata API stage can only call stored procedures that have all IN parameters or all INOUT and OUT parameters, and the number of parameters

must match the number of columns on the Columns tab. A stored procedure that has all IN parameters must be invoked from the stage's input link. (This is similar to calling a target procedure in the Stored Procedure stage.) A stored procedure that has all OUT parameters must be invoked from the stage's output link. (This is similar to calling a source procedure in the Stored Procedure stage.) A stored procedure that has INOUT and OUT parameters must be invoked from the stage's reference link. The INOUT parameters must have the Key attribute checked on the Columns tab, and the OUT parameters must not have the Key attribute checked. Calling a stored procedure from the reference link or output link requires Teradata API version 1.2.5 or later.

The ODBC stage supports calling Teradata macros and stored procedures, but it does not support calling them on a reference link.

The Stored Procedure stage will not have the restrictions of the Teradata API and ODBC stages, so it will be the preferred method for calling Teradata macros, stored procedures, scalar functions, and table functions. And since the Stored Procedure stage will allow calling macros and table functions that return more than one row for each input row, this essentially gives the user the ability to do a cursor lookup. The ability to do a cursor lookup into Teradata is currently not available in any other InfoSphere DataStage components.

The following table lists the recommended InfoSphere DataStagecomponent for each situation.

Use Case
Recommended Connectivity Stage

Call a Teradata macro
Stored Procedure

Call a Teradata stored procedure
Stored Procedure

Call a Teradata scalar user-defined function
Stored Procedure

Call a Teradata table user-defined function
Stored Procedure

Call a Teradata aggregate user-defined function
Teradata API

Import parameters to the Stored Procedure stage Parameters tab
Stored Procedure Definitions import

Import macro output parameters to the Stored Procedure or Teradata API Columns tab
Stored Procedure Definitions import

Import macro input parameters, stored procedure parameters, or function parameters to the Stored Procedure or Teradata API Columns tab
Teradata API stage import

The following features are specific to Teradata.

Importing Stored Procedures

About this task

If you intend to import a stored procedure instead of a macro, you must select **Disable CALL to EXEC Conversion** on the **Teradata ODBC Driver Option** screen. Also, you should set **Session Mode** to the transaction mode that was used to compile the stored procedure.

Stored procedures and user-defined functions have predefined schemas. Therefore, all their parameters are imported into the **Parameters** tab with their **I/O Type** set appropriately. The **RETURN** type indicates that the parameter is the return value of a scalar function or aggregate function. Only one parameter will have a type of **RETURN**.

Importing Macros

When importing a macro, the import facility displays the macro's input parameters. A macro's output schema can only be determined by executing the macro, so the import facility gives you the option of supplying an argument list. Clicking **Cancel** does not execute the macro, so only the input parameters are imported.

If the macro is executed during import, the **Columns** tab of the **Table Definition** displays the macro's output columns. The **Parameters** tab displays the macro's input parameters.

Options for Client Character Set

The options for **Client character set** are:

- Default
- Auto
- ASCII (Unix and Windows)
- EBCDIC (IBM)
- EBCDIC037_0E (IBM U.S. and Canada)
- EBCDIC273_0E (IBM Austria and Germany)
- EBCDIC277_0E (IBM Denmark and Norway)
- HANGULEBCDIC933_1II (Korean IBM)
- HANGULKSC5601_2R4 (Korean Unix and Windows)
- KANJIEBCDIC5026_0I (Japanese IBM)
- KANJIEUC_0U (Japanese Unix)
- KANJISJIS_0S (Japanese Windows)
- LATIN1_0A (ISO 8859-1 Latin 1 Unix)
- LATIN9_0A (ISO 8859-15 Latin 9 Unix)
- LATIN1252_0A (Latin Windows)
- SCHGB2312_1T0 (Simplified Chinese Unix and Windows)
- CHBIG5_1R0 (Traditional Chinese Unix and Windows)
- TCHEBCDIC937_3IB (Traditional Chinese IBM)
- UTF8 (Universal Transformation Format)

If **Database vendor** is a job parameter, the Teradata-specific client character sets do not appear in the list of available characters sets. Only **Default** and **Auto** appear in

the list. However, it is still possible to specify a Teradata-specific client character set or a job parameter in the field. This field has no effect for other vendors.

Default specifies that the installed Teradata client's default character set should be used to connect to the Teradata server.

Auto specifies that the Teradata client character set should be set automatically based on the stage map on the **NLS** tab as shown in the following table.

Server Job NLS Stage Map	Parallel Job NLS Stage Map	Teradata Client Character Set
ASCII	ASCL_ASCII	ASCII
BIG5	ASCL_BIG5	TCHBIG5_1R0
EBCDIC	ASCL_EBCDIC	EBCDIC
EBCDIC-037	ASCL_EBCDIC-037	EBCDIC037_0E
EBCDIC-1026	ASCL_EBCDIC-1026	KANJIEBCDIC5026_0I
	IBM273	EBCDIC273_0E
	IBM277	EBCDIC277_0E
EBCDIC-IBM933	ASCL_EBCDIC-IBM933	HANGULEBCDIC933_1II
EBCDIC-IBM937	ASCL_EBCDIC-IBM937	TCHEBCDIC937_3IB
GB2312	ASCL_GB2312	SCHGB2312_1T0
ISO8859-1	ASCL_ISO8859-1	LATIN1_0A
ISO8859-15	ASCL_ISO8859-15	LATIN9_0A
JPN-EUC	ASCL_JPN-EUC	KANJIEUC_0U
KSC5601	ASCL_KSC5601	HANGULKSC5601_2R4
MS1252	ASCL_MS1252	LATIN1252_0A
SHIFT-JIS	ASCL_SHIFT-JIS	KANJISJIS_0S
UTF8	UTF-8	UTF8

ASCII, EBCDIC, and UTF8 are permanently installed Teradata client character sets and are always available. The other client character sets must be installed on the Teradata server to be usable.

Error Codes

You can specify Teradata error codes be treated as fatal errors or warnings just as they can for other database vendors. If you treat an error as a warning in Teradata transaction mode, note that previous input rows can be affected by the error. If an error occurs in Teradata transaction mode, Teradata automatically rolls back the current transaction. In ANSI transaction mode, an error does not affect the current transaction. If an error in Teradata transaction mode causes uncommitted rows to be rolled back, the Stored Procedure stage issues an Info message in the Director log that specifies the number of affected input rows. The Stored Procedure stage also issues an Info message for Teradata's rollback error message, "RDBMS code 3514: User-generated transaction ABORT." This message can be changed to a warning or fatal error message by putting 3514 in the warnings or fatal errors list.

Activity Count to Link

When selected, the number of rows affected by the macro or procedure is sent down the output link. A macro returns the number of rows affected by the

statements in the macro, such as the number of rows inserted, updated, deleted, and selected. A stored procedure that has only IN parameters always returns a count of 0. A scalar user-defined function or a stored procedure that has INOUT or OUT parameters returns a count of 1, or it returns 0 if there was an error. A table user-defined function returns the number of rows returned.

Returning Multiple Rows

When **Procedure returns multiple rows** is cleared, only the first row is returned from macros and table functions that return multiple rows. This is similar to doing a singleton lookup. When it is selected, each call to a macro or table function can send multiple rows down the output link. This is equivalent to doing a cursor lookup. If a macro returns multiple result sets, the number of columns in each result set must match the number of output parameters on the Parameters tab, and the columns must be data type compatible with the output parameters.

If **Database procedure type** is **Stored procedure** or **Scalar user-defined function**, **Procedure returns multiple rows** is not active, since those objects cannot return multiple rows.

If the executed object is a macro and **Procedure returns multiple rows** is selected, one or more rows are sent down the output link for each statement in the macro. Each SELECT statement in the macro can send multiple rows down the output link. Other statements each send a row down the output link. In each output link row, the procedure status and activity count columns contain the status and count for that statement.

If **Procedure returns multiple rows** is cleared, the status columns contain the status of the first statement in the macro, and the activity count column contains the sum of the activity counts from each statement in the macro.

Managing Cursors

Procedure EOD return code is not active if **Database vendor** is **Teradata**. When a Teradata stored procedure returns, it closes any cursors that it opened, so this facility is not useful to fetch multiple rows from Teradata stored procedures. To fetch multiple rows, use a macro or a table user-defined function.

SQL Server

The following features are specific to SQL Server.

Scalar User-Defined Functions

When **Database procedure type** on the **Syntax** tab is **Scalar user-defined function** (see **"The Syntax Tab"**), the return value must be the first parameter in the value parameters grid on the **Parameters** tab with the **Parameter type** set to "Function" (see **"The Parameters Tab"**).

Returning Result Sets and Output Parameters

When the stage has an output link and the **Procedure type** is **Source** (see **"The Syntax Tab"**),

- If the executed procedure returns a result set, **Procedure returns multiple rows** must be selected (see **"The General Tab"**).
- If the executed procedure returns output parameters, **Procedure returns multiple rows** must be cleared.

For source procedures, there is no way to return a result set and output parameters, because a parameter type does not exist. Use a **Procedure type** of **Transform** to accomplish this.

Return Value

Since the Stored Procedures stage handles the return value using **Procedure status to link** (see **"The General Tab"**), when you click **Load** (see **"Columns Tab"**), you must manually remove or delete the first parameter named "Return Value."

Output Parameters and Return Codes

Output parameters and return codes do not appear in the data stream until all results have been fetched. For this reason, an extra row in the output contains the result code and output parameter values, with every other column containing nulls. Every other row has the return code and output parameter set to zero (0) and null respectively. This extra row can be filtered out using a transform to check for the return code from the procedure.

Procedure Call Syntax and Return Codes

The procedure call syntax always uses a return code `{?=call...}` even if you do not select **Procedure status to link** (see **"The General Tab"**).

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:

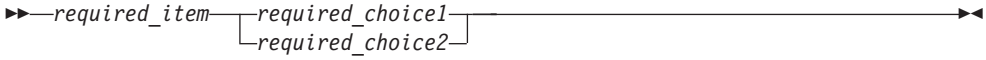
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol 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.

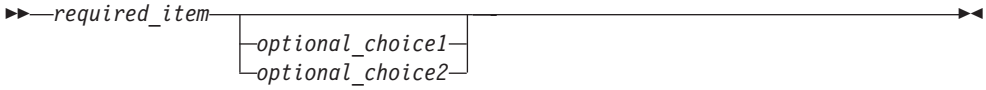
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

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

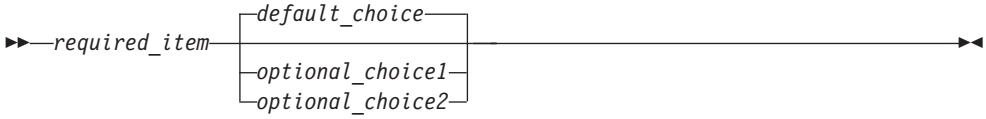
  If you must choose one of the items, one item of the stack appears on the main path.



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

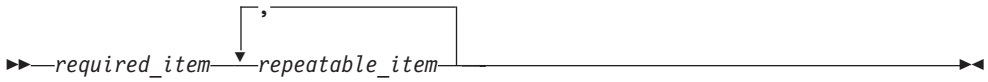


  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



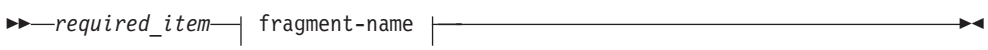
- An arrow returning to the left, above the main line, indicates an item that can be repeated.

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 9. 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 10. 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 10. 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

W

Printed in USA

SC19-4268-00