

Ascential DataStage™

DB2/UDB API Stage Guide

Version 1.2



This document, and the software described or referenced in it, are confidential and proprietary to Ascential Software Corporation ("Ascential"). They are provided under, and are subject to, the terms and conditions of a license agreement between Ascential and the licensee, and may not be transferred, disclosed, or otherwise provided to third parties, unless otherwise permitted by that agreement. No portion of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Ascential. The specifications and other information contained in this document for some purposes may not be complete, current, or correct, and are subject to change without notice. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT, INCLUDING WITHOUT LIMITATION STATEMENTS REGARDING CAPACITY, PERFORMANCE, OR SUITABILITY FOR USE OF PRODUCTS OR SOFTWARE DESCRIBED HEREIN, SHALL BE DEEMED TO BE A WARRANTY BY ASCENTIAL FOR ANY PURPOSE OR GIVE RISE TO ANY LIABILITY OF ASCENTIAL WHATSOEVER. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL ASCENTIAL BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs.

© 2004, 1999-2003 Ascential Software Corporation. All rights reserved. DataStage®, EasyLogic®, EasyPath®, Enterprise Data Quality Management®, Iterations®, Matchware®, Mercator®, MetaBroker®, Application Integration, Simplified®, Ascential™, Ascential AuditStage™, Ascential DataStage™, Ascential ProfileStage™, Ascential QualityStage™, Ascential Enterprise Integration Suite™, Ascential Real-time Integration Services™, Ascential MetaStage™, and Ascential RTI™ are trademarks of Ascential Software Corporation or its affiliates and may be registered in the United States or other jurisdictions.

Adobe Acrobat is a trademark of Adobe Systems, Inc. DB2, DB2 Universal Database, and IBM, are either registered trademarks or trademarks of IBM Corporation. Microsoft, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd. Other marks mentioned are the property of the owners of those marks.

The software delivered to Licensee may contain third-party software code. See *Legal Notices* ([LegalNotices.pdf](#)) for more information.

How to Use This Guide

The DB2/UDB API stage reads data from an IBM DB2 Universal Database (UDB) and writes it to any Ascential DataStage stage, and it reads data from any Ascential DataStage stage and writes it to an IBM DB2 database. It also provides native data browsing and meta data importing from the DB2 database to Ascential DataStage. Version 1.2 of Teradata Load is compatible with Ascential DataStage Release 7.5.1.

Audience

This guide is intended for DataStage designers who create or modify jobs that use the DB2/UDB API stage.

How This Book is Organized

The following table lists topics that may be of interest to you and it provides links to these topics.

To learn about	Read...
Functionality	"Functionality" on page 2
Installation	"Installing the Plug-In" on page 3
Defining the DB2 connection	"Defining the DB2 Connection" on page 4
Defining DB2 input data	"Defining DB2 Input Data" on page 7
Writing data to DB2	"Writing Data to DB2" on page 12
Importing meta data	"Importing Meta Data" on page 14
Defining DB2 output data	"Defining DB2 Output Data" on page 15
Reading data from DB2	"Reading Data from DB2" on page 18
Data type support	"Data Type Support" on page 19

Related Documentation

To learn more about documentation from other Ascential products and third-party documentation as they relate to DB2/UDB API, refer to the following sections/tables.

Ascential Software Documentation

Guide	Description
<i>Ascential DataStage Server Job Developer's Guide</i>	Instructions for using a stage in a DataStage job
<i>Ascential DataStage Designer Guide</i>	General principles for designing jobs
<i>Ascential MetaStage User's Guide</i>	Information about Ascential MetaStage™
<i>Ascential DataStage NLS Guide</i>	Information about NLS and techniques for character-set mapping
<i>Ascential DataStage Plug-In Installation and Configuration Guide</i>	Information required to configure your system and install this stage
<i>DataStage Enterprise Edition: Parallel Job Developer's Guide</i>	Detailed discussion of SQL Builder

DB2/UDB Documentation

Guide	Description
<i>DB2 Connect Quick Beginnings Guide</i>	Information about the configuration of client/server connections

Conventions

Convention	Used for...
bold	Field names, button names, menu items, and keystrokes. Also used to indicate filenames, and window and dialog box names.
<code>user input</code>	Information that you need to enter as is.
<code>code</code>	Code examples

Convention	Used for...
<i>variable</i> or <variable>	Placeholders for information that you need to enter. Do not type the greater-/less-than brackets as part of the variable.
>	Indicators used to separate menu options, such as: Start >Programs >Ascential DataStage
[A]	Options in command syntax. Do not type the brackets as part of the option.
B...	Elements that can repeat.
A B	Indicator used to separate mutually-exclusive elements.
{ }	Indicator used to identify sets of choices.

Contacting Support

To reach Customer Care, please refer to the information below:

Call toll-free: 1-866-INFONOW (1-866-463-6669)

Email: support@ascentialsoftware.com

Ascential Developer Net: <http://developernet.ascential.com>

Please consult your support agreement for the location and availability of customer support personnel.

To find the location and telephone number of the nearest Ascential Software office outside of North America, please visit the Ascential Software Corporation website at <http://www.ascentialsoftware.com>.

Contents

Audience	iii
How This Book is Organized	iii
Related Documentation	iv
Ascential Software Documentation	iv
DB2/UDB Documentation	iv
Conventions	iv
Contacting Support	v
Introduction	1
Functionality	2
Installing the Plug-In	3
Defining the DB2 Connection	4
Connecting to a DB2 Data Source	5
Defining Character Set Mapping	6
Defining DB2 Input Data	7
About the Input Page	7
Reject Row Handling	11
Writing Data to DB2	12
Importing Meta Data	14
Defining DB2 Output Data	15
About the Output Page	16
Reading Data from DB2	18
Data Type Support	19
Mapping Data Types from DataStage SQL to DB2 SQL	20
Mapping Data Types from DB2 SQL to DataStage SQL	21
Handling \$ and # Characters	22
Troubleshooting	23

Introduction

Before this stage, you could only access DB2 data from your DataStage jobs by using the ODBC stage. The ODBC stage offers you full access to data in DB2 databases through an ODBC driver manager and a DB2-specific ODBC driver. Though this solution is acceptable to many, others may find the generic ODBC interface inadequate for specifying database operations that are unique to DB2 and for importing DB2 meta data. It is also inadequate for situations where DB2 behavior is nonstandard. For example, users of Ascential DataStage on UNIX platforms often need to obtain additional third-party software and configure ODBC data sources.

This plug-in processes SQL statements in the native DB2 environment. It also provides native importing of meta data definitions into the Ascential DataStage Repository as well as live data browsing during job design.

The DB2 plug-in enables Ascential DataStage to write data to and read data from a DB2 database. The DB2 plug-in is a passive stage that can have any number of input, output, and reference output links.

- **Input links.** Specify the data you are writing, which is a stream of rows to be loaded into a DB2 database. You can specify the data on an input link using an SQL statement generated by Ascential DataStage or constructed by the user.
- **Output links.** Specify the data you are extracting, which is a stream of rows to be read from a DB2 database. You can specify the data on an output link using an SQL SELECT statement generated by Ascential DataStage or constructed by the user.
- **Reference output links.** Represent rows that are key read from a DB2 database using the key columns in a WHERE clause of the SELECT statement. These statements can be constructed by Ascential DataStage or specified by the user. The key columns are determined by column definitions specified for the link.

In summary, the purpose of this plug-in is to eliminate the need for the ODBC stage in order to access DB2 data by providing native capabilities for the following:

- Reading and writing data (DML)
- Creating and dropping tables (DDL)
- Importing table and column definitions (meta data)
- Browsing native data with the custom DB2 property editor

For more information on using a plug-in stage in a job, see *Ascential DataStage Server Job Developer's Guide*.

Functionality

The DB2 plug-in has the following functionality:

- Supports Versions 7 and 8 of DB2.
- Connects to DB2 on an AS/400 system using Client Application Enablers (CAEs).
- Uses stream input, stream output, and reference output links.
- Imports table and column definitions from the target DB2 database and stores them in the DataStage Repository. For more information about meta data import, see *Ascential DataStage Server Job Developer's Guide*.
- Automatically generates SQL statements to read or write DB2 data. (You can override this with user-defined SQL statements.)
- Automatically drops and creates specified target tables. (You can override this with user-defined SQL statements.)
- Uses file names to contain your SQL statements.
- Provides a custom user interface for editing the DB2 plug-in properties.
- Uses stored procedures.
- Supports NLS (National Language Support). For information, see *Ascential DataStage NLS Guide*.
- Allows data browsing through the custom property editor. You can use the custom GUI for the plug-in to view sample native table data residing on the target DB2 database.
- Supports Ascential MetaStage™. For information, see *Ascential MetaStage User's Guide*.
- Supports reject row handling. For information, see *Ascential DataStage Server Job Developer's Guide*.

The following functionality is not supported:

- Bulk loading of DB2 tables from stream input. Although vast amounts of data may be read into a DB2 database using this plug-in, the stream input links are not designed for performance-critical loading. You should use the DB2/UDB Load stage for this purpose.
- Replacing the ODBC stage. The DB2 plug-in does not replace the ODBC stage. The ODBC stage will continue to exist for access to data for which Ascential DataStage does not provide a native interface. Users who created jobs using the ODBC stage to access a DB2 database may continue to run these jobs.
- The large object family of DB2 data types (BLOB and DBCLOB).

Installing the Plug-In

The DB2 server plug-in requires Client Application Enablers (CAEs) if the DB2 data resides on an AS/400 system.

On the DataStage server machine, set the following system environment variables:

- **PATH.** Set to include the path to the DB2 *bin* directory.
- **DB2 PATH.** Set to the installation directory of DB2.

Any changes to system environment variables require a system reboot before the values of the variables take effect.

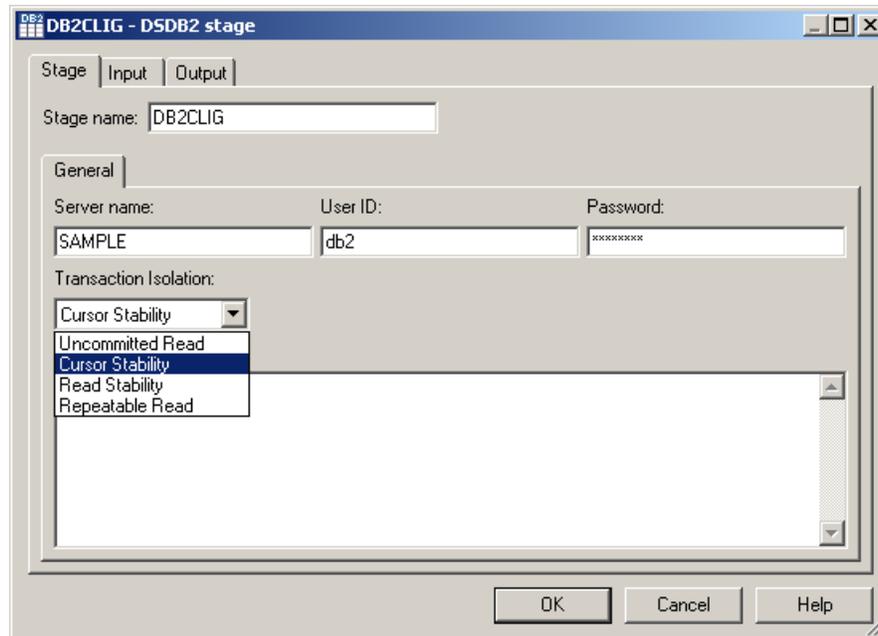
For instructions and information supporting the installation, see *Ascential DataStage Plug-In Installation and Configuration Guide*.

You can temporarily use the grid-style editor by right-clicking the plug-in icon and choosing *Grid Style* from the shortcut menu. Use this editor if you want to use job parameters for property values. (You cannot use the GUI to enter job parameters in boxes that require numeric values.)

For more information about the stage editor, see *Ascential DataStage Designer Guide*. For information on using a stage in a job, see *Ascential DataStage Server Job Developer's Guide*.

Defining the DB2 Connection

Using the plug-in GUI is easier than using grids to edit the values they contain. When you use the plug-in GUI to edit a DB2 stage, the **DSDDB2 Stage** dialog box appears:



This dialog box has **Stage**, **Input**, and **Output** 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 DB2 server name, login information, and transaction isolation level information for concurrency control in jobs. You can describe the purpose of the stage in the **Description** field. The properties on this page define the connection to the DB2 data source. For details, see ["Connecting to a DB2 Data Source"](#) on page 5.

The **NLS** tab defines a character set map to be used with the stage. This tab appears only if you have installed NLS for Ascential DataStage. For details, see ["Defining Character Set Mapping"](#) on page 6.

- Input.** This page is displayed only if you have an input link to this stage. It specifies the SQL table to use and the associated column definitions for each data input link. It also specifies how data is written and contains the SQL statement or call syntax used to write data to a DB2 table. It also specifies how to create the target table if desired and how to drop it if necessary.

- **Output.** This page is displayed only if you have an output or reference output link to this stage. It specifies the SQL tables to use and the associated column definitions for each data output link. It contains the SQL SELECT statement or call syntax used to read data from one or more DB2 tables or views.

The main phases in defining a DB2 stage from the **DSDB2 Stage** dialog box are:

- 1 Connect to a DB2 data source (see [page 5](#)).
- 2 Optional. Define a character set map (see [page 6](#)).
- 3 Define the data on the input links (see [page 7](#)).
- 4 Define the data on the output links (see [page 15](#)).

Click **OK** to close this dialog box. Changes are saved when you save the job design.

Connecting to a DB2 Data Source

The DB2 connection parameters are set on the **General** tab of the **Stage** page. To connect to a DB2 data source:

- 1 Enter the name of the DB2 server to access in the **Server name** field. Use the Client Configuration Assistant on Windows (Windows 2000 or Windows Server 2003) or the command line processor in UNIX to configure the DB2 server on the DB2 client, which is the DataStage server. See *DB2 Connect Quick Beginnings Guide* for more information about the configuration of client/server connections. This is a required field with no default.
- 2 Optionally, enter the name to use to connect to the DB2 database in the **User ID** field. (Without a user name in this field, the database uses the user name of the process running the DataStage job to connect to the server.)

This user must have sufficient privileges to access the specified database and source and target tables.

- 3 Enter the optional password that is associated with the specified user name to use in the **Password** field. For security, it displays asterisks instead of the value you enter. There is no default. If User ID is omitted, this field is ignored.
- 4 Choose an appropriate transaction isolation level to use from the **Transaction Isolation** list. These levels provide the necessary concurrency control between transactions in the job and other transactions. You cannot edit this field, which is required. Use one of the following transaction isolation levels:

Uncommitted Read. Takes exclusive locks on modified data. This level is equivalent to read uncommitted. 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.

Cursor Stability. Takes exclusive locks on modified data and sharable locks on all other data. This is the default. This level is equivalent to read committed. 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.

Read Stability. Identical to repeatable read except that phantom rows may be seen.

Repeatable Read. Equivalent to serializable. This level takes exclusive locks on modified data and sharable locks on all other 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.

The DB2 terminology for transaction isolation levels differs from ANSI terminology. Therefore, the options differ from those found in other plug-ins such as the Informix CLI stage.

- 5 Optionally, describe the purpose of the DB2 stage in the **Description** field.

Defining Character Set Mapping

You can define a character set map for a plug-in stage. Do this from the **NLS** tab that appears on the **Stage** page. The **NLS** tab appears only if you have installed NLS.

The default character set map is defined for the project or the job. You can change the map by selecting a map name from the list.

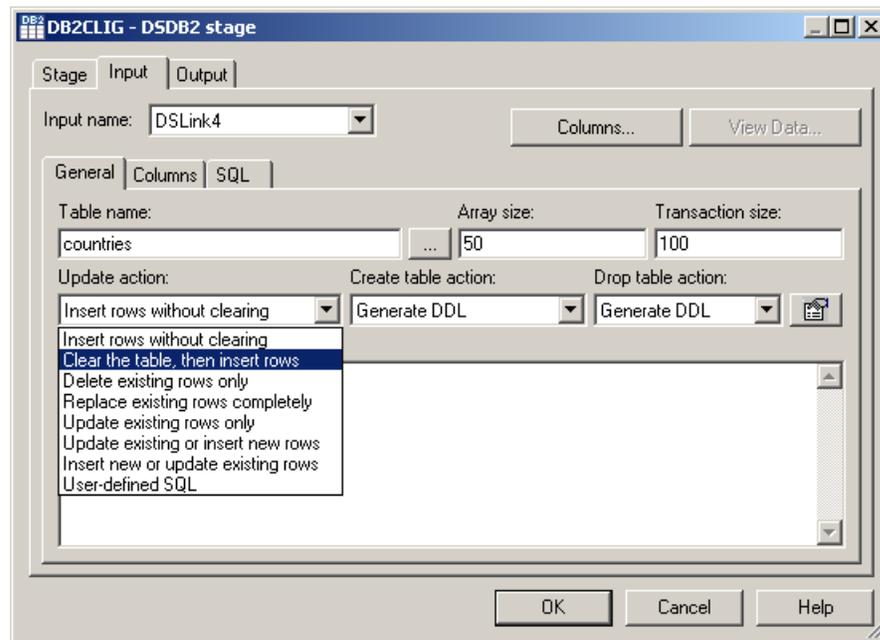
Click **Use Job Parameter...** to specify parameter values for the job. Use the format *#Param#*, where *Param* is the name of the job parameter. The string *#Param#* is replaced by the job parameter when the job is run.

Show all maps lists all the maps that are shipped with Ascential DataStage. **Loaded maps only** lists only the maps that are currently loaded.

For additional information about NLS, see *Ascential DataStage NLS Guide*. For additional information about job parameters, see *Ascential DataStage Server Job Developer's Guide*.

Defining DB2 Input Data

When you write data to a table in a DB2 database, the DB2 stage has an input link. Define the properties of this link and the column definitions of the data on the **Input** page in the **DSDB2 Stage** dialog box.



About the Input Page

The **Input** page has an **Input name** field, the **General**, **Columns**, and **SQL** tabs, and the **Columns...** and **View Data...** buttons:

- **Input name.** The name of the input link. Choose the link you want to edit from the **Input name** drop-down list box. This list displays all the input links to the DB2 stage.
- Click the **Columns...** button to display a brief list of the columns designated on the input link. As you enter detailed meta data in the **Columns** tab, you can leave this list displayed.
- Click the **View Data...** button to start the Data Browser. This lets you look at the data associated with the input link. For a description of the Data Browser, see *Ascential DataStage Server Job Developer's Guide*.

General Tab

This tab is displayed by default. It contains the following fields:

- **Table name.** This field is editable when **Update action** is *not* **User-defined SQL** (otherwise, it is read-only). It is the name of the target table to update. You must specify the target table if you do not specify **User-defined SQL**. There is no default. You can also click the ... button at the right of **Table name** to browse the Repository to select the table.
- **Array size.** The input parameter array size. The default is 50 rows that are cached before being written to the database. The array size value should be an integer greater than or equal to 1.
- **Transaction size.** The number of rows that the plug-in processes before committing a transaction to the database. The transaction size should always be a multiple of the array size. The default is 100.

The transaction size should be an integer greater than or equal to 0. A value of 0 means that the transaction will not be committed until all rows have been processed.

- **Update action.** Specifies which stage-generated SQL statements are used to update the target table. Some update actions require key columns to update or delete rows. The default is **Insert rows without clearing**. Choose one of the following options:
 - **Insert rows without clearing.** Inserts the new rows in the table.
 - **Clear the table, then insert rows.** Deletes the contents of the table before inserting the new rows.
 - **Delete existing rows only.** Deletes existing rows in the target file that have identical keys in the input rows.
 - **Replace existing rows completely.** Deletes the existing rows, then adds the new rows to the table.
 - **Update existing rows only.** Updates the existing data rows. Any rows in the data that do not exist in the table are ignored.
 - **Update existing or insert new rows.** Updates the existing data rows before inserting new rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows exist in the target table, it is faster to update first.
 - **Insert new or update existing rows.** Inserts the new rows before updating existing rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows do not exist in the target table, it is faster to insert first.

Note If using **Update existing or insert new rows** or **Insert new or update existing rows** for the update action, **Array size** must be 1. (Otherwise, a warning is logged and the Plug-in automatically sets it to 1.)

- **User-defined SQL.** Writes the data using a user-defined SQL statement. When you select this option, it overrides the default SQL statement generated by the stage. If you choose this option, you enter data on the **SQL** tab. See "[Using User-Defined SQL Statements](#)" on page 13 for details on how to do this.

Note If a table that is being updated is also being used for reference lookups, **Array size** must be 1 so the updates can be referenced.

- **Create table action.** Choose one of the following options to create the target table in the specified database:
 - **Do not create target table.** Specifies that the target table is not created, and the **Drop table action** field and the **Table Properties** button (at the right of the field) are disabled. If the target table does not exist when you run a job, the job aborts.
 - **Generate DDL.** Specifies that the stage generates the CREATE TABLE statement using information obtained from the **Table name** field, the column definitions grid, and the advanced table properties (see the description for the **Table Properties** button later in this section). If the target table already exists, the job aborts.
 - **User-defined DDL.** Specifies that you enter the appropriate CREATE TABLE statement on the **SQL** tab described on [page 10](#). You can customize the stage-generated DDL that the stage provides as a template. If the target table already exists, the job aborts.
- **Drop table action.** Lets you control the dropping of the target table before it is created by the stage. If you choose not to create the target table, this field is disabled. Choose one of the following options:
 - **Do not drop target.** Specifies that the target table is not dropped.
 - **Generate DDL.** Specifies that the stage generates DDL based on the value of the **Table name** field. If the target table does not exist, a warning is logged. The job does not abort.
 - **User-defined DDL.** Specifies that you should define the DDL to drop the target table. You can customize the stage-generated DDL that the stage provides as a template. If the target table does not exist, a warning is logged. The job does not abort.

- **Table Properties button.** Click the button at the right of the **Drop table action** list box to display the **Create Table Properties** dialog box. (This button is enabled when you select **Generate DDL** or **User-defined DDL** from the **Create table action** list box.) You can then specify the following advanced table properties from this dialog box.
 - **Tablespace.** Specifies an existing tablespace name. The new table is created in this tablespace. If you omit the name, the table is created in the default tablespace as defined by the database.
 - **Partitioning Key.** Specifies the columns to use for partitioning the data for a table in a multipartitioned nodegroup. If you omit this field and the table resides in a multipartitioned nodegroup, the table is partitioned using the default partitioning rules as defined by the database.

Columns Tab

This tab contains the column definitions for the data written to the table or file. The **Columns** tab behaves the same way as the **Columns** tab in the ODBC stage. For a description of how to enter and edit column definitions, see *Ascential DataStage Server Job Developer's Guide*.

SQL Tab

This tab contains the following tabs. Use these tabs to display the stage-generated SQL statement and the SQL statement that you can enter.

- **Generated.** Displays the SQL statements constructed by Ascential DataStage that are used to write data to DB2. The statements represent the uneditable result of the selection made in the **Update action** field on the **General** tab. You can use **Copy** to copy them to the Clipboard for use elsewhere. This tab is displayed by default.
- **User-defined.** Select **User-defined SQL** from the **Update action** field on the **General** tab to enable this tab. The GUI displays the stage-generated SQL statement on this tab as a starting point. However, you can enter any valid, appropriate SQL statement. You can scroll or change the box size proportionately to display long SQL statements when you resize the main window. For more information, see "[Using User-Defined SQL Statements](#)" on page 13.
- **Before.** This tab contains the SQL statements executed before the stage processes any job data rows. The elements on this tab correspond to the Before SQL and Continue if Before SQL fails

grid properties. The **Before** and **After** tabs look alike. The Continue if Before SQL fails property is represented by a check box and the SQL statement is entered in a resizable edit box.

- **After.** This tab contains the SQL statements executed after the stage processes job data rows. The elements on this tab correspond to the After SQL and Continue if After SQL fails grid properties. The **Before** and **After** tabs look alike. The Continue if After SQL fails property is represented by a check box and the SQL statement is entered in a resizable edit box.
- **Generated DDL.** Select **Generate DDL** or **User-defined DDL** from the **Create table action** field on the **General** tab to enable this tab. The **CREATE statement** field displays the uneditable CREATE TABLE statement that is generated from the column meta data definitions and the information provided on the **Create Table Properties** dialog box. If you select an option other than **Do not drop target table** from the **Drop table action** list, the **DROP statement** field displays the generated DROP TABLE statement for dropping the target table.
- **User-defined DDL.** Select **User-defined DDL** from the **Create table action** or **Drop table action** field on the **General** tab to enable this tab. The generated DDL statement is displayed as a starting point from which you can define a CREATE TABLE and a DROP TABLE statement.

The **DROP statement** field is disabled if **User-defined DDL** is not selected from the **Drop table action** field. If **Do not drop target** is selected, the **DROP statement** field is empty in the **Generated DDL** and **User-defined DDL** tabs.

Note Once you modify the user-defined DDL statement from the original generated DDL statement, changes made to other table-related properties do not affect the user-defined DDL statement. If, for example, you add a new column in the column grid after modifying the user-defined DDL statement, the new column appears in the generated DDL statement but does not appear in the user-defined DDL statement. You must ensure that the user-defined SQL results in the creation or dropping of the correct target table.

Reject Row Handling

During input link processing, rows of data may be rejected by the database for various reasons, such as unique constraint violations or data type mismatches.

The DB2 stage writes the offending row to the log for the DataStage job. The log, however, does not identify the row value that caused the row to be rejected. You must use the error messages returned by the DB2 database to identify it.

Ascential DataStage provides additional reject row handling. To use this capability:

- 1 Set the Parameter Array Size property to 1.
- 2 Use a Transformer stage to redirect the rejected rows.

You can then design your job by choosing an appropriate target for the rejected rows, such as a Sequential stage. You can then reuse this target as an input source once you resolve the issues with the offending row values.

Writing Data to DB2

The following sections describe the differences when you use stage-generated or user-defined SQL INSERT, DELETE, or UPDATE statements to write data from Ascential DataStage to a DB2 database.

Using Generated SQL Statements

By default, Ascential DataStage writes data to a DB2 table using an SQL INSERT, DELETE, or UPDATE statement that it constructs. The generated SQL statement is automatically constructed using the DataStage table and column definitions that you specify in the input properties for this stage. The **Generated** tab on the **SQL** tab displays the SQL statement used to write the data.

To use a generated statement:

- 1 Enter table names in the **Target name** field on the **General** tab of the **Input** page.
- 2 Specify how you want the data to be written by choosing an option from the **Update action** list. See "[General Tab](#)" on page 8 for a description of the update actions.
- 3 Enter an optional description of the input link in the **Description** field.
- 4 Click the **Columns** tab on the **Input** page.
- 5 Edit the Columns grid to specify column definitions for the columns you want to write.

The SQL statement is automatically constructed using your chosen update action and the columns you have specified. You can now optionally view this SQL statement.

- 6 Click the **SQL** tab on the **Input** page, then the **Generated** tab to view this SQL statement. You cannot edit the statement here, but you can access this tab at any time to select and copy parts of the generated statement to paste into the user-defined SQL statement.
- 7 Click **OK** to close this dialog box. Changes are saved when you save your job design.

The following table summarizes the GUI update actions and the corresponding values in the property grid.

GUI Update Action	Property Grid Value
Insert rows without clearing	insert
Clear the table, then insert rows	clear then insert
Delete existing rows only	delete
Replace existing rows completely	delete then insert
Update existing rows only	update
Update existing or insert new rows	update or insert
Insert new or update existing rows	insert or update
User-defined SQL	N/A. Replaces the User Defined SQL Statement property. It also implicitly replaces the Generate SQL property. If you choose this option, enter data on the SQL tab, described on page 10 .

Using User-Defined SQL Statements

Instead of writing data using an SQL statement constructed by Ascential DataStage, you can enter your own SQL INSERT, DELETE, or UPDATE statement for each DB2 input link. Ensure that the SQL statement contains the table name, the type of update action you want to perform, and the columns you want to write.

To enter an SQL statement:

- 1 Choose **User-defined SQL** from the **Update action** list on the **General** tab of the **Input** page. The **SQL** tab appears.
- 2 Click the **User-defined** tab on the **SQL** tab. The **User-defined** tab appears.

By default you see the stage-generated SQL statement. You can edit this statement or enter the SQL statement you want to use to write data to the target DB2 tables. This statement must contain the table name, the type of update action you want to perform, and the columns you want to write.

If the property value begins with {FILE}, the remaining text is interpreted as a pathname, and the contents of the file supplies the property value.

When writing data, the INSERT statements must contain a VALUES clause with parameter markers (?) for each stage input column. UPDATE statements must contain a SET clause with parameter markers for each stage input column. UPDATE and DELETE statements must contain a WHERE clause with parameter markers for the primary key columns. The parameter markers must be in the same order as the associated columns listed in the stage properties. For example:

```
INSERT emp (emp_no, emp_name) VALUES (?, ?)
```

If you specify multiple SQL statements, each is executed as a separate transaction. End individual SQL statements with a semicolon (;).

The size of this box changes proportionately when the main window is resized in order to allow the convenient display of very long and/or complex SQL statements.

Unless you specify a user-defined SQL statement, the stage automatically generates an SQL statement.

- 3 Click **OK** to close this dialog box. Changes are saved when you save your job design.

Importing Meta Data

To import table definitions from the DB2/UDB API stage using the Ascential DataStage Manager:

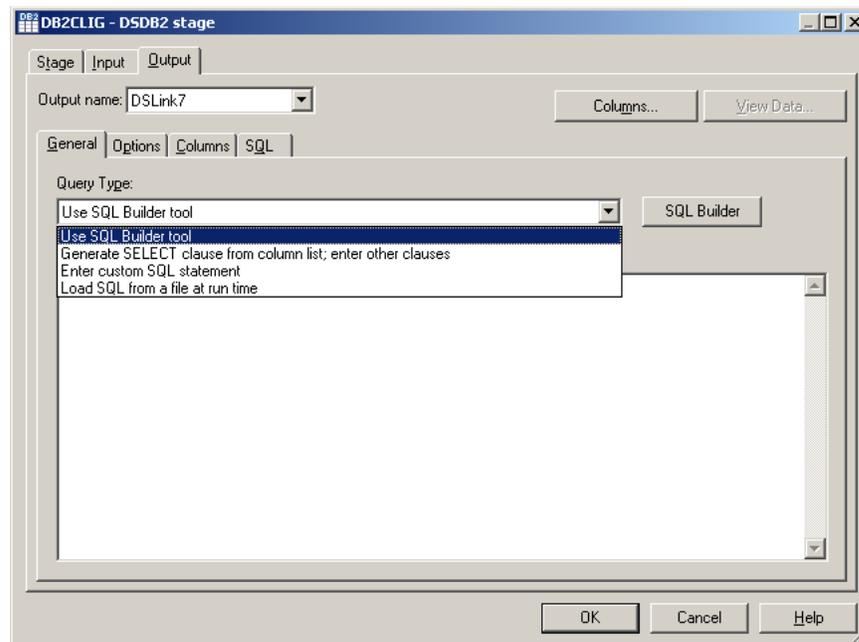
- 1 Choose **Import > Table Definitions > Plug-in Meta Data Definitions...** . The **Import Plug-in Meta Data** dialog box appears, listing the plug-ins for which import meta data facilities are available:
- 2 Choose the DSDB2 plug-in, then click **OK**. This opens the **Import DSDB2 Meta Data** wizard-style dialog that guides you through the rest of the import process.
- 3 Select the desired database objects to be imported. You can select your own tables, views, system tables, and aliases. You can also view these objects by their fully qualified names. Click **Next** to display the table list from the selected server:
- 4 When you have selected the tables from which to import meta data and specified the destination category, click **Import**. If you select a single table from the list, you can view the list of columns

in the table or view its contents by clicking **Detail...** or **View Data...** respectively. These features let you browse the list of tables before deciding whether to import the meta data into the Ascential DataStage Repository. Or, you can use **Select all** to import meta data from all the tables.

For more information about importing table definitions, see *Ascential DataStage Manager Guide* and *Ascential DataStage Server Job Developer's Guide*.

Defining DB2 Output Data

When you read data from a DB2 data source, the DB2 stage has an output link. The properties of this link and the column definitions of the data are defined on the **Output** page in the **DSDB2 Stage** dialog box.



About the Output Page

The **Output** page has an **Output name** field, the **General**, **Options**, **Columns**, and **SQL** tabs, and the **Columns...** and **View Data...** buttons. The tabs displayed depend on how you specify the SQL statement to output the data.

- **Output name.** The name of the output link. Choose the link you want to edit from the **Output name** drop-down list box. This list displays all the output links.
- Click the **Columns...** button to display a brief list of the columns designated on the output link. As you enter detailed meta data in the **Columns** tab, you can leave this list displayed.
- Click the **View Data...** button to start the Data Browser. This enables you to look at the data associated with the output link. For a description of the Data Browser, see *Ascential DataStage Server Job Developer's Guide*.

General Tab

This tab is displayed by default. It provides the type of query and, where appropriate, a button to open an associated dialog box. The **General** tab contains the following field:

- **Query type.** Displays the following options.
 - **Use SQL Builder Tool.** Specifies that the SQL statement is built using the SQL Builder graphical interface. When this option is selected, the **SQL Builder** button appears. If you click **SQL Builder**, the SQL Builder opens. See *DataStage Enterprise Edition: Parallel Job Developer's Guide* for a complete description of the SQL Builder. This is the default setting.
 - **Generate SELECT clause from column list; enter other clauses.** Specifies that DataStage generates the SELECT clause based on the columns you select on the **Columns** tab. When this option is selected, the **SQL Clauses** button appears. If you click **SQL Clauses**, the **SQL Clauses** dialog box appears. Use this dialog box to refine the SQL statement.
 - **Enter custom SQL statement.** Specifies that a custom SQL statement is built using the **SQL** tab. See "[SQL Tab](#)" on page 17.
 - **Load SQL from a file at run time.** Specifies that the data is extracted using the SQL query in the pathname of the designated file that exists on the server. Enter the pathname for this file instead of the text for the query. With this choice, you can edit the SQL statements.

- **Description.** Lets you enter an optional description of the output link.

Options Tab

This tab provides the number of prefetch rows.

- **Prefetch rows.** The number of rows that DB2 returns when DataStage fetches data from the source tables. Specifying a value greater than 1 improves performance (memory usage increases to accommodate buffering multiple rows). The value should be an integer greater than or equal to 1.

Columns Tab

This tab contains the column definitions for the data being output on the chosen link. For a description of how to enter and edit column definitions, see *Ascential DataStage Server Job Developer's Guide*.

Enter the appropriate table name in the **Description** field on output links to qualify column references. Do this if any ambiguity exists as to which table the indicated columns belong.

The column definitions for reference links require a key field. Key fields join reference inputs to a Transformer stage. The DB2 plug-in key reads the data by using a WHERE clause in the SQL SELECT statement.

SQL Tab

This tab displays the stage-generated or user-defined SQL statements or stored procedure call syntax used to read data from DB2. It contains the **Query**, **Before**, and **After** tabs.

- **Query.** This tab is read-only if you select **Use SQL Builder tool** or **Generate SELECT clause from column list; enter other clauses** for **Query Type**. If **Query Type** is **Enter Custom SQL statement**, this tab contains the SQL statements executed to read data from DB2. The GUI displays the stage-generated SQL statement on this tab as a starting point. However, you can enter any valid, appropriate SQL statement. If **Query Type** is **Load SQL from a file at run time**, enter the path name of the file.
- **Before.** This tab contains the SQL statements executed before the stage processes any job data rows.
- **After.** This tab contains the SQL statements executed after the stage processes all job data rows.

Reading Data from DB2

The following sections describe the differences when you use the SQL Builder, generated queries or user-defined queries to read data from a DB2 database into Ascential DataStage.

Using the SQL Builder

The SQL Builder provides a graphical interface that helps you build SQL SELECT statements. These statements allow you to select rows of data from your relational database. The statement can be a simple one that selects rows from a single table, or it can be complex, performing joins between multiple tables or aggregations of values within columns.

The SQL Builder is fully documented in *DataStage Enterprise Edition: Parallel Job Developer's Guide*.

- To use the SQL Builder:
 - 1 Select **Use SQL Builder tool** from the **Query type** list on the **General** tab of the **Output** page. The **SQL Builder** button appears. Click **SQL Builder**. The SQL Builder appears.
 - 2 Follow the instructions documented in *DataStage Enterprise Edition: Parallel Job Developer's Guide*.

Using Generated Queries

Ascential DataStage extracts data from a DB2 data source using an SQL SELECT statement that it constructs. The SQL statement is automatically constructed using the table and column definitions that you entered in the stage output properties.

When you select **Generate SELECT clause from column list; enter other clauses**, data is extracted from a DB2 database using an SQL SELECT statement constructed by Ascential DataStage. Also the **SQL Clauses** button appears. Click **SQL Clauses**. The **SQL Clauses** dialog box appears.

SQL SELECT statements have the following syntax:

```
SELECT clause FROM clause
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause];
```

When you specify the tables to use and the columns to be output from the DB2 stage, the SQL SELECT statement is automatically

constructed and can be viewed by clicking the **SQL** tab on the **Output** page.

For example, if you extract the Name, Address, and Phone columns from a table called Table1, the SQL statement displayed on the **SQL** tab is:

```
SELECT Name, Address, Phone FROM Table1;
```

The **SELECT** and **FROM** clauses are the minimum required and are automatically generated by Ascential DataStage. However, you can use any of these SQL **SELECT** clauses:

SELECT clause	Specifies the columns to select from the database.
FROM clause	Specifies the tables containing the selected columns.
WHERE clause	Specifies the criteria that rows must meet to be selected.
GROUP BY clause	Groups rows to summarize results.
HAVING clause	Specifies the criteria that grouped rows must meet to be selected.
ORDER BY clause	Sorts selected rows.

For more information about these clauses, see *Ascential DataStage Server Job Developer's Guide*.

Using User-Defined Queries

Instead of using the SQL statement constructed by Ascential DataStage, you can enter your own SQL statement for each DB2 output link.

- To use user-defined queries:
 - 1 Select **Enter custom SQL statement** from the **Query type** list on the **General** tab of the **Output** page. The **SQL** tab is enabled.
 - 2 You can edit the statements or drag and drop the selected columns into your user-defined SQL statement. You must ensure that the table definitions for the output link are correct and represent the columns that are expected.
 - 3 If your entry begins with {FILE}, the remaining text is interpreted as a pathname, and the contents of the file supplies the text for the query.

- 4 Click **OK** to close this dialog box. Changes are saved when you save your job design.

Data Type Support

The following sections show you the mapping from DataStage SQL data types to DB2 SQL data types and the mapping from DB2 SQL data types to the DataStage SQL types.

Mapping Data Types from DataStage SQL to DB2 SQL

When the Create Table property is set to Yes for input links, the target table is created using the column definitions for the input link and the specific input link properties defining the target table's properties. In some cases, there is no exact translation between a DB2 data type and a DataStage data type, for example, GRAPHIC.

The following table shows the DB2 data types that are generated from the corresponding DataStage types:

DataStage SQL Data Type	DB2 SQL Data Type
SQL_BIGINT	BIGINT
SQL_BINARY	CHAR FOR BIT DATA
SQL_BIT	Unsupported
SQL_CHAR	CHAR
SQL_DATE	DATE
SQL_DECIMAL	DECIMAL
SQL_DOUBLE	DOUBLE PRECISION
SQL_FLOAT	FLOAT
SQL_INTEGER	INTEGER
SQL_LONGVARIABLE	LONG VARCHAR FOR BIT DATA
SQL_LONGVARCHAR	LONG VARCHAR
SQL_LONGVARCHAR	CLOB (see note below)
SQL_NUMERIC	DECIMAL
SQL_REAL	REAL
SQL_SMALLINT	SMALLINT
SQL_TIME	TIME

DataStage SQL Data Type	DB2 SQL Data Type
SQL_TIMESTAMP	TIMESTAMP
SQL_TINYINT	SMALLINT
SQL_VARBINARY	VARCHAR FOR BIT DATA
SQL_VARCHAR	VARCHAR

Note The DB2/UDB API Plug-in supports the CLOB data type by mapping the LONGVARCHAR data type with a precision greater than 32 K to DB2 UDB's CLOB data type. To work with a CLOB column definition, choose DataStage's LONGVARCHAR as the column's data type and provide a Length of more than 32 K in the **Columns** tab. If the Length is less than or equal to 32 K, DataStage's LONGVARCHAR maps to LONGVARCHAR.

Mapping Data Types from DB2 SQL to DataStage SQL

Conversely, when the DB2 stage imports meta data definitions from a DB2 database, it must perform a mapping of the DB2 SQL data types to the SQL data types supported by Ascential DataStage. The following table describes the mapping between the DB2 SQL data types and the DataStage SQL data types:

DB2 SQL Data Type	DataStage SQL Data Type
BIGINT	SQL_BIGINT
CHAR	SQL_CHAR
CHAR FOR BIT DATA	SQL_BINARY
DATE	SQL_DATE
DECIMAL	SQL_DECIMAL
DOUBLE PRECISION	SQL_DOUBLE
FLOAT	SQL_FLOAT
GRAPHIC	SQL_CHAR
INTEGER	SQL_INTEGER
LONG VARCHAR	SQL_LONGVARCHAR
LONG VARCHAR FOR BIT DATA	SQL_LONGVARBINARY
LONG VARGRAPHIC	SQL_LONGVARCHAR
NUMERIC	SQL_NUMERIC

DB2 SQL Data Type	DataStage SQL Data Type
REAL	SQL_REAL
SMALLINT	SQL_SMALLINT
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
VARCHAR FOR BIT DATA	SQL_VARBINARY
BLOB and LOCATOR	Unsupported
BLOB and LOCATOR	Unsupported
CLOB and LOCATOR	SQL_LONGVARCHAR (see note below)
DBCLOB and LOCATOR	Unsupported

Note The DB2/UDB API Plug-in supports the CLOB data type by mapping the LONGVARCHAR data type with a precision greater than 32 K to DB2 UDB's CLOB data type. To work with a CLOB column definition, choose DataStage's LONGVARCHAR as the column's data type and provide a Length of more than 32 K in the **Columns** tab. If the Length is less than or equal to 32 K, DataStage's LONGVARCHAR maps to LONGVARCHAR.

Handling \$ and # Characters

Ascential DataStage has been modified to enable it to handle DB2 databases which use the reserved characters # and \$ in column names. DataStage converts these characters into an internal format, then converts them back as necessary.

To take advantage of this facility, do the following:

- In Ascential DataStage Administrator, open the **Environment Variables** dialog box for the project in question, and set the environment variable DS_ENABLE_RESERVED_CHAR_CONVERT to true (this can be found in the General\Customize branch).
- Avoid using the strings __035__ and __036__ in your DB2 column names (these are used as the internal representations of # and \$ respectively).

Import meta data using the Plug-in Meta Data Import tool; avoid hand-editing (this minimizes the risk of mistakes or confusion).

Once the table definition is loaded, the internal column names are displayed rather than the original DB2 names both in table definitions and in the Data Browser. They are also used in derivations and expressions. The original names are used in generated SQL statements, however, and you should use them if entering SQL in the job yourself.

When using a DB2 Plug-In in a server job, you should use the external names when entering SQL statements that contain DB2 columns. The columns within the stage are represented by question marks (parameter markers) and bound to the DB2 columns by order, so you do not need to worry about entering names for them. This applies to:

- Query
- Update
- Insert
- Key
- Select
- Where clause

For example, for an update you might enter:

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

Particularly note the key in this statement ($\$A\#$) is specified using the external name.

Troubleshooting

If your source data is defined correctly, rows are properly inserted in a target table. However, under certain conditions rows may not be inserted in a target table. DB2 rejects the remainder of the row batch following a 'bad' row when the following three conditions occur:

- The Array Size property exceeds 1.
- The defined string lengths of source data exceed the defined length of its target column.
- The source data contains a row with a character string that exceeds the length of the target column.

Erroneous error messages concerning those remaining rows also result.

Example

Suppose the target table defines a column as CHAR(5), the DataStage meta data for this column is defined as CHAR(10), and the source data contains the following rows:

```
'ABC'  
'ABCD'  
'ABCDEFGH' (Longer than 5 characters)  
'AB'  
'ABCD'
```

The last three rows are not inserted into the target table when the Array Size property is set to 5. DB2 reports that all three rows contained values that were too large (DB2 error SQL0302N).

Additionally, using BIGINT for source data that contains out-of-range values causes similar behavior.

Solutions

Define the DataStage meta data correctly to match the DB2 target, and ensure that any BIGINT source data is within BIGINT range. Otherwise, it may be safer to run the job with Array Size set to 1. However, this can impact performance.

Another solution is to use a Transformer stage to scrub the data before sending it to the DB2 stage. This method also impacts performance.

Note This behavior does not occur for rows rejected by the database for reasons such as constraint violations or nonnull violations. The remaining rows in a batch are not rejected.