

Ascential **DataStage**

ODBC Enterprise Stage Reference Guide
Version 1.0

Part No. 00D-001ES10
December 2004

This document, and the software described or referenced in it, is 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 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.

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

Overview of the ODBC Enterprise Stage

Introduction

The *ODBC Enterprise Stage Reference Guide* gives a detailed description of the operators underlying the DataStage Enterprise Edition ODBC stage. It is intended for users who are familiar with the OSH language utilized by the DataStage parallel engine.

DataStage can share data with external data sources. This data can be accessed by the ODBC stage operating in one of four modes:

- In read mode the stage reads records from an external data source table and places them in an DataStage data set. The `odbcread` operator is used for this.
- In write mode the stage sets up a connection to an external data source and inserts records into a table. The Stage takes a single input data set. The write mode determines how the records of a data set are inserted into the table. The `odbcwrite` operator is used for this.
- In upsert mode the stage lets you insert data into an external data source table or update an external data source table with data contained in a DataStage data set. You can match records based on field names and then update or insert those records. The `odbcupsert` operator is used for this.
- In lookup mode the stage lets you perform a join between an external data source table and a DataStage data set, with the resulting data output as a DataStage data set. The `odbclookup` operator is used for this.

Accessing External Data Source from DataStage

This section assumes that DataStage users have been configured to access external data source(s), using the external data source configuration process.

! To access External data source from DataStage:

1. When Enterprise Edition is on a distributed DataStage Server and the parallel engines shares the same ODBC settings:
The DataDirect branded ODBC drivers will be installed in the directory `$dshome/./branded_odbc`. The shared library path will be modified to include `$dshome/./branded_odbc/lib`. The ODBCINI environment variable will be set to `$dshome/.odbc.ini`
2. Start External data source.
3. Add `$APT_ORCHHOME/branded_odbc` to your PATH and `$APT_ORCHHOME/branded_odbc/lib` to your LIBPATH, LD_LIBRARY_PATH, or

SHLIB_PATH. The ODBCINI environment variable must be set to the full path of the odbc.ini file.

4. Should access the external data source using a valid user name and password.

National Language Support

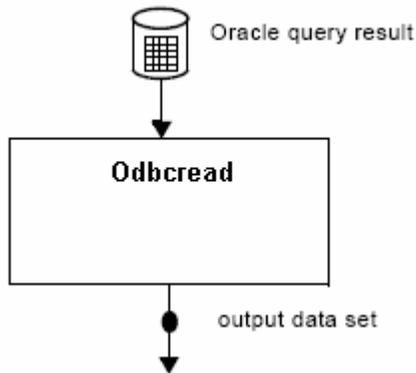
DataStage's National Language Support (NLS) makes it possible to process data in international languages using Unicode character sets. NLS is built on IBM's International Components for Unicode (ICU).

The DataStage External data source stages support Unicode character data in schema, table, and index names; in user names and passwords; column names; table and column aliases; SQL*Net service names; SQL statements; and file-name and directory paths.

The ODBC Stage in Read Mode

Underlying the ODBC stage operating in read mode is the odbcread operator. The **odbcread** operator sets up a connection to an external data source table, using an SQL query to request rows (records) from the table, and outputs the result as a DataStage data set.

Data Flow Diagram



Properties

Table **odbcread** operator Properties

Property	value
Number of input data sets	0
Number of output data sets	1
Input interface schema	None
Output interface schema	Determined by the SQL query
Transfer behavior	None
Execution mode	Sequential

Partitioning method	Not applicable
Collection method	Not applicable
Preserve-partitioning flag in output dataset	Clear
Composite Stage	no

ODBC Read

The **odbcread operator** performs basic reads from a data source. Since it relies on ODBC interfaces to connect and to import data from a data source, it does not have the flexibility to perform database reads in parallel. This Stage can only be run in a sequential mode. The options are as follows:

Options	Value
-data_source	data_source_name Specify the data source to be used for all database connections. This option is required.
-user	user_name Specify the user name used to connect to the data source. This option may <i>not be required</i> depending on the data source
-password	password Specify the password used to connect to the data source. This option may <i>not be required</i> depending on the data source
-tablename	table_name Specify the table to be read from. May be fully qualified. The -table option is mutually exclusive with the -query option. This option has 2 suboptions: -filter where_predicate Optionally specify the rows of the table to exclude from the read operation. This predicate will be appended to the where clause of the SQL statement to be executed. -selectlist select_predicate Optionally specify the list of column names that will appear in the select clause of the SQL statement to be executed.
-open	open_command Optionally specify an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.
-close	close_command

Optionally specify an SQL statement to be executed after the insert array is processed. You cannot commit work using this option. The statements are executed only once on the conductor node.

`-query` `sql_query`

Specify an SQL query to read from one or more tables. The `-query` option is mutually exclusive with the `-table` option.

`-fetcharraysize` `n`

Specify the number of rows to retrieve during each fetch operation Default is 1.

`-isolation_level` `read_uncommitted | read_committed | repeatable_read | serializable`

Optionally specify the isolation level for accessing data. The default isolation level is decided by the database or possibly specified in the data source.

`-db_cs` `code_page`

Optionally specify the ICU code page which represents the database character set in use. The default is ISO-8859-1. This option has the following sub option:

`-use_strings`

If this option is set, strings (instead of ustrings) will be generated in the DataStage schema.

Operator Action

Here are the chief characteristics of the odbcread operator:

- You can direct it to run in specific node pools.
- It translates the query's result set (a two-dimensional array) row by row to a DataStage data set.
- Its output is a DataStage data set that you can use as input to a subsequent DataStage Stage.
- Its translation includes the conversion of external data source datatypes to DataStage datatypes.
- The size of external data source rows can be greater than that of DataStage records.
- The Stage specifies either an external data source table to read or to perform an SQL query.

- It optionally specifies commands to be run before the read operation is performed and after it has completed the operation.
- You can perform a join operation between DataStage dataset and external data source (there may be one or more tables) data.

Where the odbcread Operator Runs

odbcread operator runs sequentially.

Column Name Conversion

An External data source result set is defined by a collection of rows and columns. The **odbcread** operator translates the query's result set (a two-dimensional array) to a DataStage data set. The External data source query result set is converted to a DataStage data set in the following way:

- The Schema of an external data source result set should match the schema of a DataStage data set.
- The columns of an external data source row should correspond to the fields of a DataStage record; the name and datatype of an external data source column should correspond to the name and datatype of a DataStage field.
- Names are translated exactly except when the external data source column name contains a character that DataStage does not support. In that case, two underscored characters replace the unsupported character.
- Both external data source columns and DataStage fields support nulls, and a null contained in an external data source column is stored as keyword NULL in the corresponding DataStage field.

Datatype Conversion

The **odbc read** operator converts external data source datatypes to OSH datatypes, as in the following table:

Mapping of ODBC datatypes to OSH datatypes

ODBC Datatype	DataStage Datatype
SQL_CHAR	string[n]
SQL_VARCHAR	string[max=n]
SQL_WCHAR	ustring(n)
SQL_WVARCHAR	ustring(max=n)
SQL_DECIMAL	decimal(p,s)
SQL_NUMERIC	decimal(p,s)

SQL_SMALLINT	in16
SQL_INTEGER	int32
SQL_REAL	decimal(p,s)
SQL_FLOAT	decimal(p,s)
SQL_DOUBLE	decimal(p,s)
SQL_BIT	int8 (0 or 1)
SQL_TINYINT	int8
SQL_BIGINT	int64
SQL_BINARY	raw(<i>n</i>)
SQL_VARBINARY	raw(max= <i>n</i>)
SQL_TYPE_DATE ^[6]	date
SQL_TYPE_TIME ^[6]	time[<i>p</i>]
SQL_TYPE_TIMESTAMP ^[6]	timestamp[<i>p</i>]
SQL_GUID	string[36]

Datatypes that are not listed in the table above generate an error.

External data source Record Size

The size of a DataStage record is limited to 32 KB. However, external data source records can be larger than 32 KB. If you attempt to read a record larger than 32 KB, DataStage returns an error and terminates the application.

Targeting the Read Operation

When reading an external data source table, you can either specify the table name that allows DataStage to generate a default query that reads the total records of the table, or you can explicitly specify the query.

Specifying the External data source Table Name

If you choose the **-table** option, DataStage issues the following SQL SELECT statement to read the table:

```
select [selectlist]
      from table_name
      and (filter);
```

You can specify optional parameters to narrow the read operation. They are as follows:

- The *selectlist* specifies the columns of the table to be read; by default, DataStage reads all columns.
- The *filter* specifies the rows of the table to exclude from the read operation; DataStage reads all rows by default.

You can optionally specify an **-open** and **-close** option command. These commands are executed by odbc on the external data source before the table is opened and after it is closed.

Specifying an SQL SELECT Statement

If you choose the **-query** option, you pass an SQL query to the Stage. The query operates (select) on the table as it is read into DataStage. The SQL statement can contain joins, views, database links, synonyms, and so on. However, the following restrictions apply to **-query**:

- The **-query** may not contain bind variables.
- If you want to include a filter or select list, you must specify them as part of the query.
- The query runs sequentially. .
- You can specify optional open and close commands. External data source runs these commands immediately before reading the data from the table and after reading the data from the table.

Join Operations

You can perform a join operation between DataStage data sets and external data source. First use the ODBC Stage and then either the **lookup** Stage or a join Stage. See the Parallel Job Developer's Guide for information about these stages.

Syntax and Options

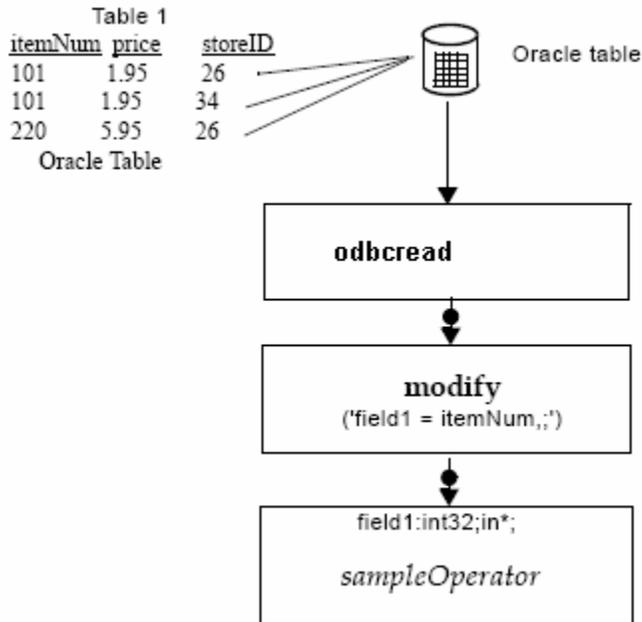
The syntax for the `odbcread` operator follows. The optional values you supply are shown in italics. When your value contains a space or a tab character, you must enclose it in single quotes.

```
odbcread
-query sql_query
-data source dsn1
-user user1
-password password1
-tablename table_name [-filter filter] [-selectlist list]
[-close close_command]
[-db_cs character_set]
[-open open_command]
-use_strings xyz
-array_size 5
-isolation_level read_committed
```

You must specify either the **-query** or **-table** option. You must also specify the **-data_source**, **user and password**.

Example 1: Reading an External data source Table and Modifying a Field Name

The following figure shows an external data source table used as input to a DataStage Stage:



The External data source table contains three columns; whose datatypes the operator converts as follows:

- *itemNum* of type NUMBER[3,0] is converted to **int32**
- *price* of type NUMBER[6,2] is converted to **decimal**[6,2]
- *storeID* of type NUMBER[2,0] is converted to **int32**

The schema of the DataStage data set created from the table is also shown in this figure. *Note that the DataStage field names are the same as the column names of the External data source table.*

However, the Stage to which the data set is passed has an input interface schema containing the 32-bit integer field *field1*, while the data set created from the External data source table does not contain a field of the same name. For this reason, the **modify** Stage must be placed between **odbcread** and **sampleStage** to translate the name of the field, *itemNum*, to the name *field1*. Following is the **osh** syntax for this example:

```

$ osh "odbcread -tablename 'table_1'
-data_source data source1
-user user1
-password user1
| modify '$modifySpec' | ... "
$ modifySpec="field1 = itemNum,;"
modify
(field1 = itemNum,;)
  
```

The ODBC Stage in Write Mode

The **ODBC Stage** in write mode uses the `odbcwrite` operator. The operator sets up a connection to an external data source and inserts records into a table. The operator takes a single input data set. The write mode of the operator determines how the records of a data set are inserted into the table.

Writing to a Multibyte Database

Specifying chars and varchars

Specify `chars` and `varchars` in bytes, with two bytes for each character. The following example specifies 10 characters:

```
create table orch_data(col_a varchar(20));
```

Specifying nchar and nvarchar2 Column Size

Specify `nchar` and `nvarchar2` columns in characters. For example, this example specifies 10 characters:

```
create table orch_data(col_a nvarchar2(10));
```

Data Flow Diagram

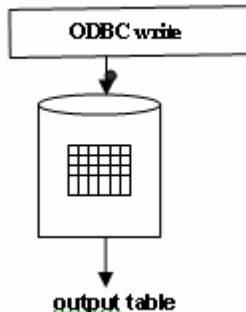


Table `odbcwrite` operator Properties

Property	Value
Number of input data sets	1
Number of output data sets	0
Input interface schema	Derived from the input data set
Output interface schema	None
Transfer behavior	None
Execution mode	Sequential default or parallel
Partitioning method	Not applicable
Collection method	Any
Preserve-partitioning flag	Default clear

Composite Stage

No

Stage Action

Here are the chief characteristics of the `odbcwrite` operator:

- Translation includes the conversion of DataStage datatypes to external data source datatypes.
- The Stage appends records to an existing table, unless you set another mode of writing
- When you write to an existing table, the input data set schema must be compatible with the table's schema.
- Each instance of a parallel write operator running on a processing node writes its partition of the data set to the external data source table. You can optionally specify external data source commands to be parsed and executed on all processing nodes before the write operation runs or after it is completed.

Where the `odbcwrite` Operator Runs

The default execution mode of the `odbcwrite` operator is parallel. The number of processing nodes is based on the configuration file by default. However, if the environment variable;

`APT_CONFIG_FILE` is set, the number of players is set to the number of nodes.

To direct the Stage to run sequentially:

Specify the `[seq]` framework argument.

You can optionally set the resource pool or a single node on which the Stage runs.

Data Conventions on Write Operations to External data source

External data source columns are named identically as DataStage fields, with these restrictions:

- External data source column names are limited to 30 characters. If an DataStage field name is longer, you can do one of the following:
 - Choose the `-truncate` or `truncate_length` options to configure the operator to truncate DataStage field names to the max., length as the data source column name. If you choose `truncate_length` then you can specify the number of characters to be truncated, and they should be less than the maximum length the data source supports
 - Use the `modify` Stage to modify the DataStage field name.
- A DataStage data set written to an external data source may not contain fields of certain types. If it does, an error occurs and the corresponding step terminates. However, DataStage offers Enterprise Edition that modifies certain datatypes to those that the

external data source accepts, as shown in the table that follows.

Datatype Conversion

Mapping of OSH datatypes to ODBC datatypes

DataStage Datatype	ODBC Datatype
string[<i>n</i>]	SQL_CHAR
string[max= <i>n</i>]	SQL_VARCHAR
ustring(<i>n</i>)	SQL_WCHAR
ustring(max= <i>n</i>)	SQL_WVARCHAR
decimal(<i>p</i> , <i>s</i>)	SQL_DECIMAL
decimal(<i>p</i> , <i>s</i>)	SQL_NUMERIC
int16	SQL_SMALLINT
int32	SQL_INTEGER
decimal(<i>p</i> , <i>s</i>)	SQL_REAL
decimal(<i>p</i> , <i>s</i>)	SQL_FLOAT
decimal(<i>p</i> , <i>s</i>)	SQL_DOUBLE
int8 (0 or 1)	SQL_BIT
int8	SQL_TINYINT
int64	SQL_BIGINT
raw(<i>n</i>)	SQL_BINARY
raw(max= <i>n</i>)	SQL_VARBINARY
date	SQL_TYPE_DATE ^[6]
time[<i>p</i>]	SQL_TYPE_TIME ^[6]
timestamp[<i>p</i>]	SQL_TYPE_TIMESTAMP ^[6]
string[36]	SQL_GUID

Write Modes

The write mode of the operator determines how the records of the data set are inserted into the destination table. The write mode can have one of the following values:

- **append:** This is the default mode. The table must exist and the record schema of the data set must be compatible with the table. The write operator appends new rows to the table. The schema of the existing table determines the input interface of the Stage.
- **create:** The operator creates a new table. If a table exists with the same name as the one being created, the step that contains the operator terminates with an error. The schema of the DataStage data set determines the schema of the new table. The table is created with simple default properties. To create a table that is partitioned, indexed, in a non-default table space, or in some other non-standard way, you can use the `-createstmt` option with your own create table statement.
- **replace:** The operator drops the existing table and creates a new one in its place. If a table exists with the same name as the one you want to create, it is overwritten. The schema of the DataStage data set determines the schema of the new table.

- **truncate:** The operator retains the table attributes but discards existing records and appends new ones. The schema of the existing table determines the input interface of the Stage. Each mode requires the specific user privileges shown in the table below:

Note: If a previous write operation failed, you can retry, specifying the **replace** write mode to delete any information in the output table that may have been written by the previous attempt to run your program.

Table **Required External data source Privileges for External data source Interface Write**

Write Mode	Required Privileges
Append	INSERT on existing table
Create	TABLE CREATE
Replace	INSERT and TABLE CREATE on existing table
Truncate	INSERT on existing table

Matched and Unmatched Fields

The schema of the External data source table determines the operator's interface schema. Once the operator determines this; it applies the following rules to determine which data set fields are written to the table:

1. Fields of the input data set are matched by name with fields in the input interface schema. DataStage performs default datatype conversions to match the input data set fields with the input interface schema.
2. You can also use the **modify** Stage to perform explicit datatype conversions.
3. If the input data set contains fields that do not have matching components in the table, the Stage generates an error and terminates the step.
4. This rule means that DataStage does not add new columns to an existing table if the data set contains fields that are not defined in the table. *Note that you can use the **odbcwrite - drop** option to drop extra fields from the data set.* Columns in the external data source table that do not have corresponding fields in the input data set are set to their default value, if one is specified in the external data source table. If no default value is defined for the external data source column and it supports nulls, it is set to null. Otherwise, DataStage issues an error and terminates the step.
5. DataStage data sets support nullable fields. If you write a data set to an existing table and a field contains a null, the External data source column must also support nulls. If not, DataStage issues an error message and terminates the step. However, you can use the modify Stage to convert a null in an input field to another value.

Syntax and Options

Syntax for the **odbcwrite** operator is given below. Optional values you supply are shown in italics. When your value contains a space or a tab character, you must enclose it in single quotes. Exactly one occurrence of the **-dboptions** option and the **-table** option are required.

```
odbcwrite
-tablename table_name
-dboptions
'{user = username, password = password}'
|
'{user = \@file_name\}'
[-close close_command]
[-createstmt create_statement]
[-drop]
[-db_cs character_set]
```

ODBCWRITE operator options

The odbcwrite operator performs basic inserts (export) to a data source. This Stage will be parallel by default. The options are as follows:

Options	Value
-data_source	<i>data_source_name</i>
	Specify the data source to be used for all database connections. This option is required.
-user	<i>user_name</i>
	Specify the user name used to connect to the data source. This option may not be required depending on the data source
-password	<i>password</i>
	Specify the password used to connect to the data source. This option may not be required depending on the data source
-tablename	<i>table_name</i>
	Specify the table to write to. May be fully qualified.
-mode	<i>append</i> <i>create</i> <i>replace</i> <i>truncate</i>
	Specify the mode for the write Stage as one of the following:
	append: new records are appended into an existing table.

create: the operator creates a new table. If a table exists with the same name as the one you want to create, the step that contains the operator terminates with an error. The schema of the new table is determined by the schema of the DataStage data set. The table is created with simple default properties. To create a table that is partitioned, indexed, in a non-default table space, or in some other non-standard way, you can use the -createstmt option with your own create table statement.

replace: The operator drops the existing table and creates a new one in its place. The schema of the DataStage data set determines the schema of the new table.

truncate: All records from an existing table are deleted before loading new records.

-createstmt create_statement

Optionally specify the create statement to be used for creating the table when – mode create is specified.

-drop

If this option is set unmatched fields of the DataStage the data set will be dropped. An unmatched field is a field for which there is no identically named field in the data source table.

-truncate

If this option is set column names are truncated to the maximum size allowed by the ODBC driver.

-truncateLength n

Specify the length to truncate column names to.

-open open_command

Optionally specify an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.

-close close_command

Optionally specify an SQL statement to be executed after the insert array is processed. You cannot commit work using this option. The statements are executed only once on the conductor node.

-insertarraysize n

Optionally specify the size of the insert array. The default size is 2000 records.

-rowCommitInterval n

Optionally specify the number of records that should be committed before starting a new transaction. This option can only be specified if arraysize = 1.

Otherwise rowCommitInterval = arraysize. This is because of the rollback logic/retry logic that occurs when an array execute fails.

-isolation_level read_uncommitted | read_committed | repeatable_read | serializable

Optionally specify the isolation level for accessing data. The default isolation level is decided by the database or possibly specified in the data source.

-db_cs code page name

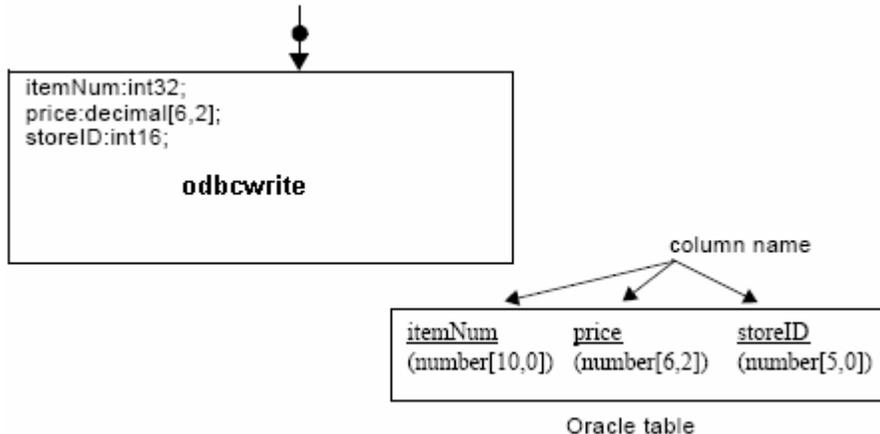
Optionally specify the ICU code page which represents the database character set in use. The default is ISO-8859-1.

Example 1: Writing to an Existing External data source Table

When an existing External data source table is written to:

- The column names and datatypes of the External data source table determine the input interface schema of the write Stage.
- This input interface schema then determines the fields of the input data set that is written to the table.

For example, the following figure shows the **odbcwrite** operator writing to an existing table:



The record schema of the DataStage data set and the row schema of the External data source table correspond to one another, and field and column names are identical. Following are the input DataStage record schema and output External data source row schema:

Input DataStage Record Output External data source Table

itemNum:int32;	price NUMBER[10,0]
----------------	--------------------

price:decimal[3,2];;	itemNum NUMBER[6,2]
storeID:int16	storeID NUMBER[5,0]

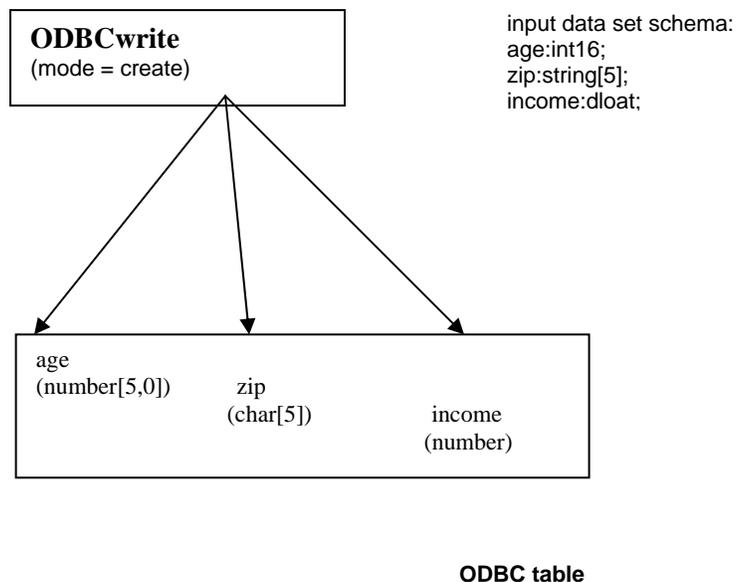
Here is the **osh** syntax for this example:

```
$ osh " ... op1 | odbcwrite -tablename 'table_2'
-data_source data sourcename -user = user101 -password passwr
```

*Note that since the write mode defaults to **append**, the **mode** option does not appear in the command.*

Example 2: Creating an External data source Table

To create a table, specify either a **create** or **replace** write mode. The next figure is a conceptual diagram of the **create** operation:



Following is the **osh** syntax for this example:

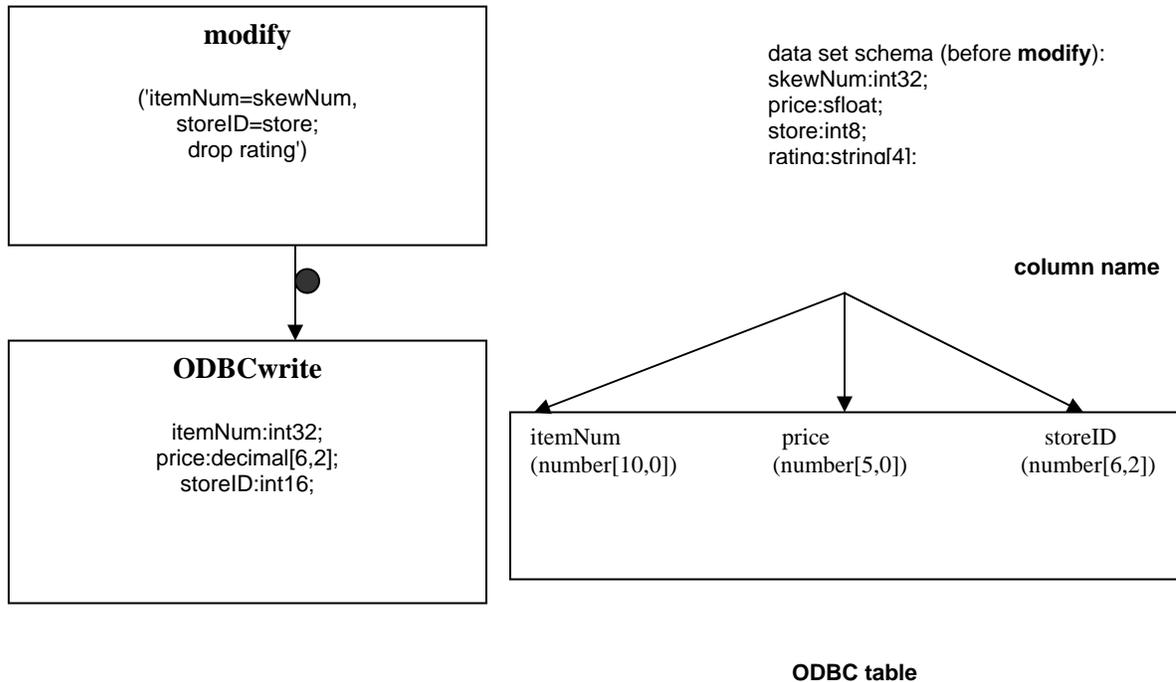
```
$ osh "... odbcwrite -table table_2
-mode create
-dboptions {'user = user101, password = userPword'} ..."
```

The **odbcwrite** operator creates the table, giving the external data source columns the same names as the fields of the input DataStage data set and converts the DataStage datatypes to external data source datatypes.

Example 3: Writing to an External data source Table Using the modify operator

The **modify** operator allows you to drop unwanted fields from the write operation and translate the name and/or datatype of a field of the input data set to match the input interface schema of the Stage.

The next example uses the **modify** operator:



In this example, the **modify** operator is used to:

- Translate field names of the input data set to the names of corresponding fields of the Stage's input interface schema, that is, *skewNum* to *itemNum* and *store* to *storeID*.
- Drop the unmatched *rating* field, so that no error occurs.

*Note that DataStage performs automatic type conversion of store, promoting its **int8** datatype in the input data set to **int16** in the **odbcwrite** input interface.*

Here is the **osh** syntax for this example:

```
$ modifySpec="itemNum = skewNum, storeID = store;drop rating"
$ osh "... op1 | modify '$modifySpec'
| odbcwrite -table table_2
-dboptions {user = user101, password =
userPword}'"
```

Other Features

Quoted Identifiers

All operators that accept SQL statements, as arguments will support quoted identifiers in those arguments. The quotes should be escaped with ‘\’ c character

Case Sensitive Column Identifiers

Because of limitations in the parallel engine, case sensitive column identifiers cannot be supported in this release. This support will be added in a future release.

Large Objects (BLOB, CLOB, etc.)

Because of limitations in the parallel engine, large objects cannot be supported in this release.

Stored Procedures

The ODBC stage will not support stored procedures. The user should use the Stored Procedure Stage for stored procedure support.

Graphical SQL Builder

The SQL builder will not be available for ODBC in this release. This support will be added in the ‘Hawk’ release or sooner.

Transaction Rollback

Because it is not possible for native transactions to span multiple processes transaction, rollback will not be possible in this release.

Unit of Work

The unit of work Stage will not be modified to support ODBC in this release.

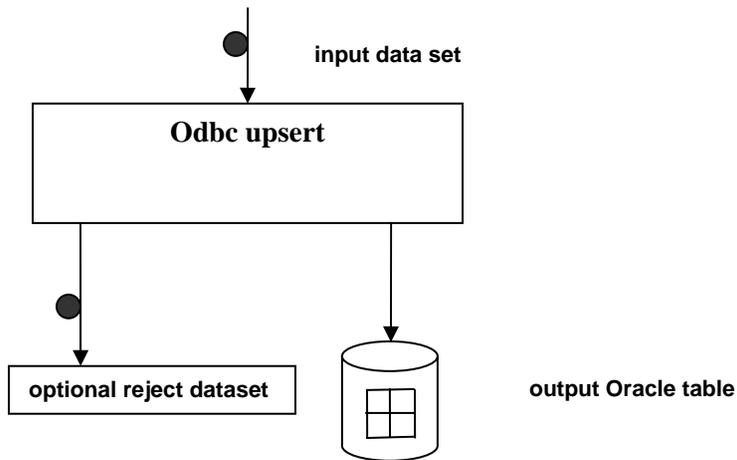
The ODBC Stage in Upsert Mode

The **ODBC stage in upsert mode uses the odbcupsert operator**. The operator inserts and updates external data source table records with data contained in a DataStage data set. You provide the insert and update SQL statements using the **-insert** and **-update** options. By default, **odbcupsert** uses external data source host-array processing to optimize the performance of inserting records.

odbcupsert

This operator receives a single data set as input and writes its output to an external data source table. You can request an optional output data set that contains the records that fail to be inserted or updated.

Data Flow Diagram



Property	Value
Number of input data sets	1
Number of output data sets by default	None; 1 when you select the -reject option
Input interface schema	Derived from your insert and update statements
Transfer behavior	Rejected update records are transferred to an output data set when you select the -reject option
Execution mode	Parallel by default, or sequential
Partitioning method	Same You can override this partitioning method; however, a partitioning method of entire cannot be used.
Collection method	Any
Combinable Stage	Yes

Operator Action

Here are the main characteristics of **odbcupsert**

- If an **-insert** statement is included, the insert is executed first. Any records that fail to be inserted because of a unique-constraint violation are then used in the execution of the update statement.
- DataStage uses host-array processing by default to enhance the performance of insert

array processing. Each insert array is executed with a single SQL statement. Update records are processed individually.

- Use the **-insertArraySize** option to specify the size of the insert array. For example:

```
-insertArraySize 250
```

- The default length of the insert array is 500. To direct DataStage to process your insert statements individually, set the insert array size to 1:

```
-insertArraySize 1
```

- The record fields can be of variable-length strings. You can either specify a maximum length or use the default maximum length of 80 characters.

This example specifies a maximum length of 50 characters:

```
record(field1:string[max=50])
```

The maximum length in this example is by default 80 characters:

```
record(field1:string)
```

- When an insert statement is included and host array processing is specified, a DataStage update field must also be a DataStage insert field.
- The **odbcupsert** operator converts all values to strings before passing them to the external data source. The following OSH datatypes are supported:

- int8, uint8, int16, uint16, int32, uint32, int64, and uint64
- dfloat and sfloat
- decimal
- strings of fixed and variable length
- timestamp
- date

- By default, **odbcupsert** produces no output data set. By using the **-reject** option, you can specify an optional output data set containing the records that fail to be inserted or updated. It's syntax is:

```
-reject filename
```

Syntax and Options

The syntax for **odbcupsert** is shown below. Optional values are shown in italics. When your value contains a space or a tab character, you must enclose it in single quotes.

```
odbcupsert  
data_source dsn -user username -password password
```

-update *update_statement*
[-insert *insert_statement*]
[-insertArraySize *n*]
[-reject]

Exactly one occurrence of the **-update** is required. All others are optional.

Specify an ICU character set to map between External data source char and varchar data and DataStage **ustring** data, and to map SQL statements for output to external data source. The default character set is UTF-8, which is compatible with the **osh** jobs that contain 7-bit US-ASCII data.

Odbcupsert Operator options

Options for the odbcupsert operator are as follows:

Options	Value
---------	-------

-data_source *data_source_name*

Specify the data source to be used for all database connections. This option is required.

-user *user_name*

Specify the user name used to connect to the data source. *This option may not be required depending on the data source*

-password *password*

Specify the password used to connect to the data source. *This option may not be required depending on the data source*

Statement options

The user must specify at least one of the following options and no more than two. An error is generated if the user does not specify a statement option or specifies more than two.

-update *update_statement*

Optionally specify the update or delete statement to be executed.

-insert *insert_statement*

Optionally specify the insert statement to be executed.

-delete *delete_statement*

Optionally specify the delete statement to be executed.

-mode *insert_update / update_insert / delete_insert*

Specify the upsert mode to be used when two statement options are specified. If only one statement option is specified, the upsert mode will be ignored.

insert_update – The insert statement is executed first. If the insert fails due to a duplicate key violation (i.e. record exists), the update statement is executed. This is the default upsert mode.

update_insert – The update statement is executed first. If the update fails because the record doesn't exist, the insert statement is executed.

delete_insert – The delete statement is executed first. Then the insert statement is executed.

-reject

If this option is set, records that fail to be updated or inserted are written to a reject data set. You must designate an output data set for this purpose. If this option is not specified, an error is generated if records fail to update or insert.

-open *open_command*

Optionally specify an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.

-close *close_command*

Optionally specify an SQL statement to be executed after the insert array is processed. You cannot commit work using this option. The statements are executed only once on the conductor node.

-insertarraysize *n*

Optionally specify the size of the insert/update array. The default size is 2000 records.

-rowCommitInterval *n*

Optionally specify the number of records that should be committed before starting a new transaction. This option can only be specified if *arraysize* = 1. Otherwise *rowCommitInterval* = *arraysize*. This is because of the rollback logic/retry logic that occurs when an array execution fails.

Example

This example updates an Oracle table that has two columns: *acct_id* and *acct_balance*, where *acct_id* is the primary key. Two of the records cannot be inserted because of unique key constraints; instead, they are used to update existing records. One record is transferred to the reject dataset because its *acct_id* generates an -error

Summarized below are the states of the Oracle table before the dataflow is run, the contents of the input file, and the action DataStage performs for each record in the input file.

Table before dataflow		Input file contents	DataStage action
Acct_id	acct_balance		
073587	45.64	873092 67.23	Update
873092	2001.89		Insert
675066	3523.62	865544 8569.23 566678 2008.56	Update
566678	89.72	678888 7888.23	Insert
		073587 82.56	Update
		995666 75.72	Insert

osh Syntax

```
$ osh "import -schema record(acct_id:string[6]; acct_balance:dfloat;)
      -file input.txt |
      hash -key acct_id |
      tsort -key acct_id |
      odbcupsert -data_source dsn -user apt -password test
      -insert 'insert into accounts
              values(DATASTAGE.acct_id,
                     DATASTAGE.acct_balance)'
      -update 'update accounts
              set acct_balance = DATASTAGE.acct_balance
              where acct_id = DATASTAGE.acct_id'
      -reject '/user/home/reject/reject.ds'"
```

Table after dataflow

acct_id	acct_balance
073587	82.56
873092	67.23
675066	3523.62
566678	2008.56
865544	8569.23
678888	7888.23
995666	75.72

The ODBC Stage in lookup Mode

The ODBC Stage in lookup mode uses the `odbclookup` operator. Using the stage in this mode, you can perform a join between one or more external data source tables and a DataStage data set. The resulting output data is a DataStage data set containing both DataStage and external data source data.

This join is performed by specifying either an SQL `SELECT` statement, or by specifying one or more External data source tables and one or more key fields on which to do the lookup.

This operator is particularly useful for *sparse lookups*, that is, where the DataStage data set you are matching is much smaller than the external data source table. If you expect to match 90% of your data, using the **odbcread** and **lookup** operators is probably more efficient.

Because **odbclookup** can do lookups against more than one external data source table, it is useful for joining multiple external data source tables in one query.

The **-statement** option command corresponds to an SQL statement of this form:

```
select a,b,c from data.testtbl
  where
    DataStage.b = data.testtbl.c
  and
    DataStage.name = "Smith"
```

The operator replaces each `DataStage.fieldname` with a field value, submits the statement containing the value to the external data source, and outputs a combination of external data source and DataStage data.

Alternatively, you can use the **-key/-table** options interface to specify one or more key fields and one or more External data source tables. The following **osh** options specify two keys and a single table:

```
-key a -key b -table data.testtbl
```

You get the same result as you would by specifying:

```
select * from data.testtbl
  where
    DataStage.a = data.testtbl.a
  and
    DataStage.b = data.testtbl.b
```

The resulting DataStage output data set includes the DataStage records and the corresponding rows from each referenced external data source table. When an external data source table has a column name that is the same as a DataStage data set field name, the External data source column is renamed using the following syntax:

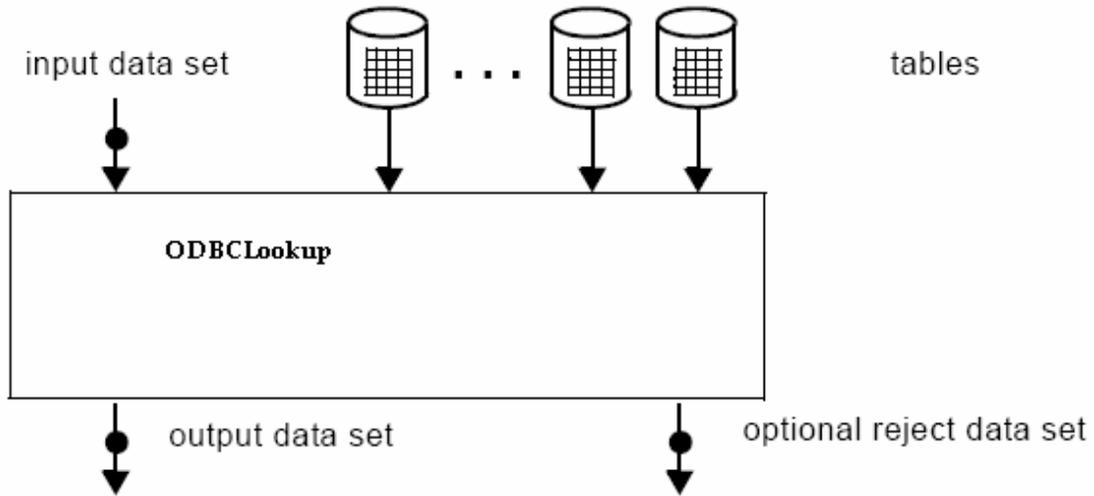
`APT_integer_fieldname`

An example is `APT_0_lname`. The *integer* component is incremented when duplicate names are encountered in additional tables.

Note If the External data source table is not indexed on the lookup keys, the performance of this

Stage is likely to be poor.

Data Flow Diagram



Properties

Table odbcllookup operator **Properties**

Property	value
Number of input data sets	1
Number of output data sets	1; 2 if you include the -ifNotFound reject option
Input interface schema	determined by the query
Output interface schema	Determined by the sql query
Transfer behavior	transfers all fields from input to output

Execution mode	sequential or parallel (default)
Partitioning method	Not applicable
Collection method	Not applicable
Preserve-partitioning flag in output dataset	Clear
Composite Stage	no

Syntax and Options

The syntax for the **odbclookup** operator is given below. Optional values are shown in italics. When your value contains a space or a tab character, you must enclose it in single quotes.

```
odbclookup
-data_source data sourcename -user username -password passwr
-tablename table_name -key field [-key field ...]
[-tablename table_name -key field [-key field ...]]
|
```

You must specify either the **-query** option or one or more **-table** options with one or more **-key** fields.

odbclookup operator Options

The odbclookup operator is parallel by default. The options are as follows:

Options	Value
-data_source <i>data_source_name</i>	Specify the data source to be used for all database connections. This option is required.
-user <i>user_name</i>	Specify the user name used to connect to the data source. <i>This option may not be required depending on the data source</i>
-password <i>password</i>	

Specify the password used to connect to the data source. *This option may not be required depending on the data source*

-tablename *table_name*

Specify a table and key fields to be used to generate a lookup query. This option is mutually exclusive with the `-query` option. The `-table` option has 3 suboptions:

-filter *where_predicate*

Specify the rows of the table to exclude from the read operation. This predicate will be appended to the where clause of the SQL statement to be executed.

-selectlist *select_predicate*

Specify the list of column names that will appear in the select clause of the SQL statement to be executed.

-key *field*

Specify a lookup key. A lookup key is a field in the table that will be used to join against a field of the same name in the DataStage dataset. The `-key` option can be specified more than once to specify more than one key field.

-ifNotFound *fail| drop | reject | continue*

Specify an action to be taken when a lookup fails. Can be one of the following:

fail: stop job execution

drop: drop failed record from the output dataset

reject: put records that are not found into a reject data set. You must designate an output dataset for this option

continue: leave all records in the output dataset (outer join)

-query *sql_query*

Specify a lookup query to be executed. This option is mutually exclusive with the `-table` option

-open *open_command*

Optionally specify an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.

-close *close_command*

Optionally specify an SQL statement to be executed after the insert array is processed. You cannot commit work using this option. The statement is executed only once on the conductor node.

-fetcharraysize *n*

Specify the number of rows to retrieve during each fetch operation. Default is 1.

-db_cs *code page name*

Optionally specify the ICU code page, which represents the database character set in use. The default is ISO-8859-1. This option has the following sub option:

-use_strings

If this option is set, strings (instead of ustrings) will be generated in the DataStage schema.

Example

Suppose you want to connect to the APT81 server as user *user101*, with the password *test*. You want to perform a lookup between a DataStage data set and a table called *target*, on the key fields *lname*, *fname*, and *DOB*. You can configure **odbclookup** in either of two ways to accomplish this.

Here is the **osh** command using the **-table** and **-key** options:

```
$ osh " odbclookup - }
```

```
-key lname -key fname -key DOB  
< data1.ds > data2.ds "
```

Here is the equivalent **osh** command using the **-query** option:

```
$ osh " odbclookup  
-query 'select * from target  
where lname = DataStage.lname  
and fname = DataStage.fname  
and DOB = DataStage.DOB'  
< data1.ds > data2.ds "
```

DataStage prints the *lname*, *fname*, and *DOB* column names and values from the DataStage input dataset and also the *lname*, *fname*, and *DOB* column names and values from the external data source table.

If a column name in the external data source table has the same name as a DataStage output data set schema fieldname, the printed output shows the column in the external data source table renamed using this format:

APT_integer_fieldname

For example, *lname* may be renamed to *APT_0_lname*.