

Ascential DataStage™

Sybase Enterprise Stage Reference Guide

Version 1.0



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.

This product or the use thereof may be covered by or is licensed under one or more of the following issued patents: US6604110, US5727158, US5909681, US5995980, US6272449, US6289474, US6311265, US6330008, US6347310, US6415286; Australian Patent No. 704678; Canadian Patent No. 2205660; European Patent No. 799450; Japanese Patent No. 11500247.

© 2005 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 Systems, Inc. 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.

Contents

Overview of the Sybase Enterprise Stage	1
Introduction	1
National Language Support	2
The sybasereade Operator	3
sybasereade	4
Operator Action	5
Where the sybasereade Runs	6
Column Name Conversion	6
Data Type Conversion	6
Targeting the Read Operation	7
Join Operations	8
Syntax and Options	8
Example 1: Reading a Sybase Table and Modifying a Field Name	9
The sybasewrite Operator	10
Writing to a Multibyte Database	10
Data Flow Diagram	11
Operator Action	11
Data Type Conversion	13
Syntax and Options	15
Example 1: Writing to an Existing Sybase Table	18
Example 2: Creating a Sybase Table	19
Example 3: Writing to a Sybase Table Using the modify Operator	20
The sybaseupsert Operator	21
Operator Action	22
Syntax and Options	23
Example	25

The sybaselookup Operator	26
Data Flow Diagram	27
Properties Table	28
Syntax and Options	28
Example	30

Overview of the Sybase Enterprise Stage

Introduction

The Sybase Enterprise stage enables sharing of data between a Sybase database and DataStage. *Sybase Enterprise Stage Reference guide* gives a detailed description of the operators underlying the DataStage Enterprise Edition's Sybase stage. It is intended for users who are familiar with the OSH language utilized by the DataStage parallel engine.

DataStage can communicate with Sybase ASE (ASE) and Sybase IQ (IQ), Relational Database Management Systems in four different modes. The Stages work uniformly across on both ASE and IQ databases and for all purposes, Sybase refers to both Sybase ASE and Sybase IQ databases. Some specific cases where there are functional and behavioral scope changes in ASE and IQ are described in the document.

- The **sybasereade** operator is used for reading from one or more tables of a Sybase database.
- The **sybasewrite** operator is used to load into a single table of a Sybase database. The write mode determines how the records of a dataset are inserted into the table.
- The **sybaseupsert** operator inserts into and/or updates and/or deletes one or more tables of a Sybase database.
- The **sybaselookup** operator is used to efficiently join large tables of a Sybase database, with small datasets.

Accessing Sybase from DataStage

To access Sybase from DataStage, follow the Sybase configuration process.

- **Sybase Client Configuration**

- Sybase open client software has to be installed on the machine for the functioning of Sybase Parallel Stage. The standard installation of the Sybase Database Server automatically installs this. Using four modes, Sybase communicates with the database using 'CT' library. For which,
 - Set your SYBASE_OCS Environment variable to the name of "Open Client" Directory in the Client Installation. This can be obtained from the server login environment as well. (For example, for Sybase 12.5 it is OCS-12_5).
 - Set your SYBASE environment variable to your Sybase installation directory.
 - If the database is Sybase ASE, set your SYBASE_ASE environment variable as it is in the server login environment (EX: For ASE 12.5 server it is ASE-12_5). If Sybase IQ server, set your ASDIR environment variable as it is in the server login environment (EX: FOR IQ 12.5 server it is ASIQ-12_5)
 - Set up the Interface file adding the details about the database server (database name, host machine name or IP address and port number)
 - Add SYBASE/bin to your PATH and &SYBASE/\$SYBASE_OCS/lib to your LIBPATH, LD_LIBRARY_PATH, or SHLIB_PATH.
 - Have login privileges to Sybase using a valid Sybase user name and a corresponding password. These must be recognized by Sybase before you attempt to access it.

Note \$SYBASE/\$SYBASE_OCS/bin must appear first in your PATH. This is to ensure that \$SYBASE/\$SYBASE_OCS/bin/isql being executed always, when user executes "isql" command.

National Language Support

DataStage's National Language Support (NLS) makes it possible for you to process data in international languages using Unicode character sets. DataStage uses International Components for Unicode (ICU) libraries to support NLS functionality. For information on National Language Support, see

<http://oss.software.ibm.com/developerworks/opensource/icu/project>

The DataStage's Sybase Parallel Enterprise Stage 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 Stage has one option which optionally control character mapping:

- `-db_cs character_set`

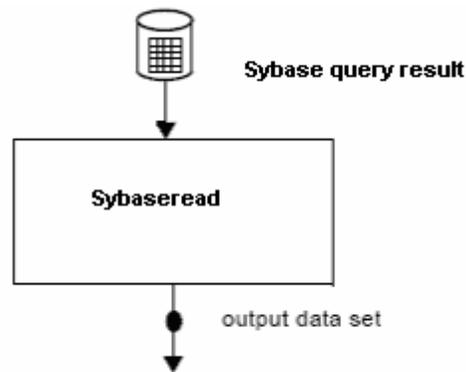
Specifies an ICU character set to map between Sybase *char* and *varchar* data and DataStage *ustring* data, and to map SQL statements for output to Sybase.

The sybasereade Operator

The sybasereade operator is used for reading from one or more tables of a Sybase database (ASE/IQ).

Data Flow Diagram

Figure 1 sybasereade



Properties

Table 1 sybasereade Properties

Property	Value
Number of input datasets	0
Number of output datasets	1
Input interface schema	None
Output interface schema	Determined by the sql query
Transfer behavior	None
Partitioning method	Not applicable

Table 1 sybasereade Properties

Property	Value
Execution mode	Sequential
Collection method	Not applicable
Preserve-partitioning flag in output dataset	Clear
Composite operator	No

sybasereade

The **sybasereade** performs basic reads from a data source. This can only be run in sequential mode. The options are as follows:

Table 2 sybasereade options

Options	Use
-db_name	database_ name Specify the database name to be used for all database connections. This option is mandatory.
-user	user_ name Specify the user name used to connect to the data source.
-password	password Specify the password used to connect to the data source.
-table	table_name[-filter filter] [-selectlist list] This option is mutually exclusive with the -query option. Specify the name of the Sybase table. The table must exist and you must have SELECT privileges on the table. The table name may contain a view name also. The -filter suboption optionally specifies a conjunction, enclosed in single quotes, to the WHERE clause of the SELECT statement to specify the rows of the table to include or exclude from reading into DataStage. The suboption -selectlist optionally specifies a SQL select list, enclosed in single quotes, that can be used to determine which fields are read.

Table 2 sybasereade options

Options	Use
-open	open_command Optionally specify a SQL statement to be executed before the insert array is processed. The statement is executed only once on the conductor node.
-close	close_command Optionally specify a SQL statement to be executed after the insert array is processed. You cannot commit work using this option. The statement is are executed only once on the conductor node.
-query	sql_query Specify a SQL query to read from one or more tables. The -query option is mutually exclusive with the -table option.
-db_cs code_page	Optionally specify the ICU code page, which represents the database character set in use. The default is ISO-8859-1.

Operator Action

Here are the chief characteristics of the sybasereade 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 dataset.
- Its output is a DataStage dataset, that you can use as input to a subsequent DataStage Stage.
- Its translation includes the conversion of Sybase data types to DataStage data types.
- The size of Sybase rows can be greater than that of DataStage records.
- The operator specifies either a Sybase table to read or a SQL query to carry out.
- 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 Sybase (there may be one or more tables) data.

Where the sybasereade Runs

The sybasereade operator runs sequentially. Sybase ASE database operator fetches all the records sequentially even in a partitioned table.

Column Name Conversion

A Sybase result set is defined by a collection of rows and columns. The sybasereade operator translates the query's result set (a two-dimensional array) to a DataStage dataset. Here is how a Sybase query result set is converted to a DataStage dataset:

- The rows of the Sybase result set correspond to the records of a DataStage dataset.
- The columns of the Sybase row correspond to the fields of a DataStage record; the name and data type of the Sybase column correspond to the name and data type of a DataStage field.
- Names are translated exactly except when the Sybase column name contains a character that DataStage does not support. In that case, two underscore characters replace the unsupported character.
- Both Sybase columns and DataStage fields support nulls, and a null contained in a Sybase column is stored as key word NULL in the corresponding DataStage field.

Data Type Conversion

The sybasereade operator converts Sybase data types to DataStage data types, as in the following table:

Table 3 Mapping of SYBASE data types to DataStage data types

SYBASE data type	Corresponding DataStage data type
tinyint	int8
smallint	int16
int	int32
integer	int32
numeric(p,s)	decimal[p,s]
decimal(p,s)	decimal[p,s]
dec(p,s)	decimal[ps,s]

Table 3 Mapping of SYBASE data types to DataStage data types

SYBASE data type	Corresponding DataStage data type
float	sfloat
double precision	dfloat
real	sfloat
smallmoney	decimal[10,4]
money	decimal[15,4]
smalldatetime	timestamp
datetime	timestamp
date	date
time	time
char(n)	string[n], a fixed-length string with length = n
varchar(n)	string[max=n], a variable-length string with maximum length = n
nchar(n)	or ustring[n], a fixed-length string with length = n, only for ASE
nvarchar(n)	ustring[max=n], a variable-length string with maximum length = n, only for ASE
binary(n)	raw(n)
varbinary(n)	raw[max=n]
bit	int8
unsigned int	uint32
money	decimal[15,4]

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

Targeting the Read Operation

When reading a Sybase 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 Sybase 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 by default, DataStage reads all rows.
- You can optionally specify an *-open* and *-close* command. These commands are executed by the sybase before the table is opened and after it is closed.

Specifying a SQL SELECT Statement

- If you choose the *-query* option, you pass a SQL query to the operator. The query specifies the table and the processing that you want to perform 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. Sybase 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 datasets and Sybase data. First invoke the sybasereade operator and then invoke either the lookup operator or a join operator. See the corresponding chapters of this manual for information on these operators. Alternatively, you can use the sybaselookup operator

Syntax and Options

The syntax for sybasereade follows. Option values you supply are shown in italic typeface. When your value contains a space or a tab character, you must enclose it in single quotes.

```
sybasereade
-query sql_query
```

```

-server servername
-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]

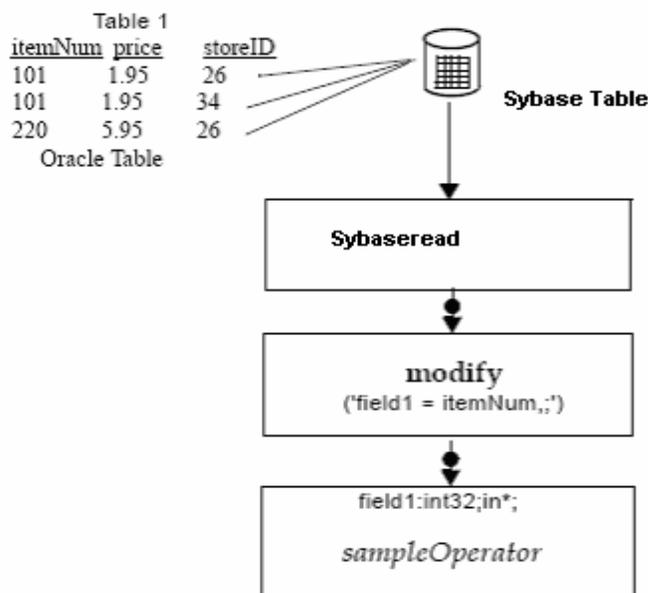
```

You may specify either the *-query* or alternatively *-table*, *-user*, and *-password* options.

Example 1: Reading a Sybase Table and Modifying a Field Name

The following figure shows a Sybase table used as input to a DataStage Stage:

Figure 2 reading & modifying Sybase table



Sybase table contains three columns, the data types in the columns are converted by the operator as follows:

- *itemNum* of type integer is converted to int32
- *price* of type NUMERIC [6,2] is converted to decimal [6,2]
- *storeID* of type integer is converted to int32

The schema of the DataStage dataset 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 Sybase table.

However, the operator to which the dataset is passed has an input interface schema containing the 32-bit integer field field1, while the dataset created from the Sybase table does not contain a field of the same name. For this reason, the modify operator must be placed between sybasereade and sample operator to translate the name of the field, itemNum, to the name field1

Here is the osh syntax for this example:

```
$ osh "sybasereade -table table_1
-server server_name
-user user1
-password user1
| modify '$modifySpec' | ... "
$ modifySpec="field1 = itemNum;"
modify
('field1 = itemNum,;')
```

The sybasewrite Operator

The sybasewrite operator sets up a connection to Sybase and inserts records into a table. The operator takes a single input dataset. The write mode of the operator determines how the records of a dataset 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. This example specifies 10 characters:

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

Data Flow Diagram

Figure 3 sybasewrite

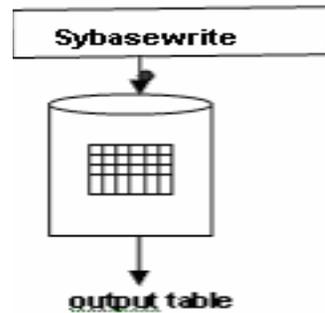


Table 4 sybasewrite Operator Properties

Property	Value
Number of input datasets	1
Number of output datasets	0
Input interface schema	Derived from the input dataset
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 operator	No

Operator Action

Here are the chief characteristics of the sybasewrite operator:

- Translation includes the conversion of DataStage data types to Sybase data types.
- The operator appends records to an existing table, unless you set another mode of writing
- When you write to an existing table, the input dataset 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 dataset to the Sybase table. You can optionally specify Sybase commands to be parsed and executed on all processing nodes before the write operation runs or after it completes.

Where the sybasewrite operator Runs

The default execution mode of the sybasewrite operator is parallel. The default is that the number of processing nodes is based on the configuration file. However, if the environment variable

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

To direct the operator to run sequentially:

"Specify the [seq] framework argument".

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

Data Conventions on Write Operations to Sybase

Sybase columns are named identically as DataStage fields, with these restrictions:

- Sybase column names are limited to 30 characters in ASE and 128 characters in IQ. If a 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 should be less than the max., length the data source supports.
 - Variable length data types *-string, raw* in DataStage is directly mapped to *varchar* and *varbinary* respectively. The default size in Sybase is 1 byte. If the operator tries to load data, which is more than 1 byte, an error message is logged. The *-stringlength* option overrides the default size of 1 byte.
 - Use the modify operator to modify the DataStage field name.
- A DataStage dataset written to Sybase may not contain fields of certain types. If it does, an error occurs and the corresponding step terminates. However DataStage offers operators that modify certain data types to ones Sybase accepts, as shown in the Table.

Data Type Conversion

Table 5 Mapping of DataStage data types to SYBASE data types

DataStage Data Type	SYBASE Data Type
uint8, int8	tinyint
int16	smallint
int32	int
int32	integer
decimal[p,s]	numeric(p,s)
decimal[p,s]	decimal(p,s)
decimal[ps,s]	dec(p,s)
sfloat	float
dfloat	double precision
sfloat	real
decimal[10,4]	smallmoney
decimal[15,4]	money
timestamp	smalldatetime
timestamp	datetime
date	date
time	time
string[n], a fixed-length string with length = n	char(n)
string[max=n], a variable-length string with maximum length = n	varchar(n)
or ustring[n], a fixed-length string with length = n	nchar(n)
ustring[max=n], a variable-length string with maximum length = n	nvarchar(n)
raw(n)	binary(n)
raw[max=n]	varbinary(n)
int8	bit
uint32	unsigned int

Write Modes

The write mode of the operator determines how the records of the dataset 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 dataset 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 operator.
- **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 DataStage dataset 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 of the same name as the one you want to create, it is overwritten. The schema of the DataStage dataset 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 operator. Each mode requires the specific user privileges shown in the table below:

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

Table 6 Sybase Privileges for Sybase Interface Write operator

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 Sybase table determines the operator's interface schema. Once the operator determines this; it applies the following rules to determine which dataset fields are written to the table:

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

Syntax and Options

Syntax for the sybasewrite operator is given below. Option 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.

```
sybasewrite


```

sybasewrite Operator Options

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

Table 7 sybasewrite options

Options	Value
-db_name	database_name Specify the data source to be used for all database connections. This option is mandatory.
-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
-server	server_name Specify the server name used to connect to the database. This option is optional.
-tablename	table_name Specify the table to write to. May be fully qualified.

Table 7 sybasewrite options

Options	Value
-mode	<p>append create replace truncate</p> <p>Specify the mode for the write operator as one of the following:</p> <ol style="list-style-type: none"> 1 append: new records are appended into an existing table. 2 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 DataStage dataset 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. 3 replace: The operator drops the existing table and creates a new one in its place. The schema of the DataStage dataset determines the schema of the new table. truncate: All records from an existing table are deleted before loading new records.
-createstmt	<p>create_statement</p> <p>Optionally specify the create statement to be used for creating the table when - mode create is specified.</p>
-drop	<p>If this option is set unmatched fields of the DataStage the dataset will be dropped. An unmatched field is a field for which there is no identically named field in the datasource table.</p>
-truncate	<p>If this option is set column names are truncated to the maximum size allowed by the SYBASE driver.</p>
-truncateLength	<p>n</p> <p>Specify the length to truncate column names to.</p>
-open	<p>open_command</p> <p>Optionally specify a SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.</p>

Table 7 sybasewrite options

Options	Value
-close	close_command Optionally specify a 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.
-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.

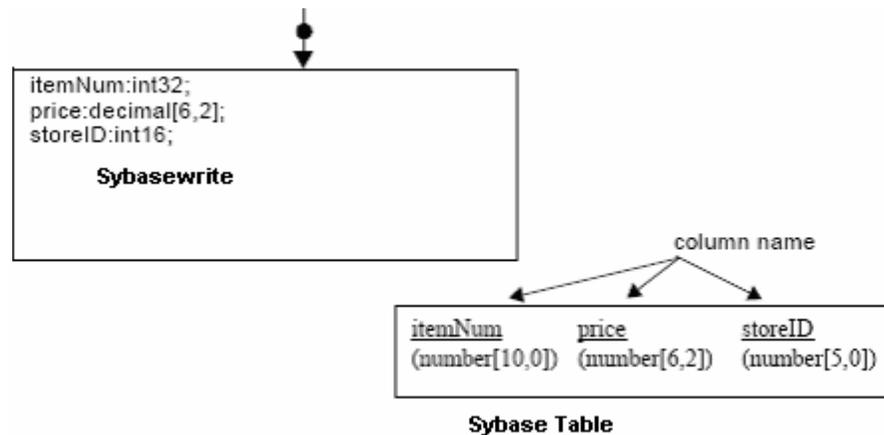
Example 1: Writing to an Existing Sybase Table

When an existing Sybase table is written to:

- The column names and data types of the Sybase table determine the input interface schema of the write operator.
- This input interface schema then determines the fields of the input dataset that is written to the table.

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

Figure 4 writing to sybase table



The record schema of the DataStage dataset and the row schema of the Sybase table correspond to one another, and field and column names are identical. Here are the input DataStage record schema and output Sybase row schema:

Table 8 comparison of record schema and row schema

Input DataStage Record	Output Sybase Table
itemNum:int32;	price Integer
price:decimal [3,2];	itemNum Decimal [3,2]
storeID:int16	storeID Small int

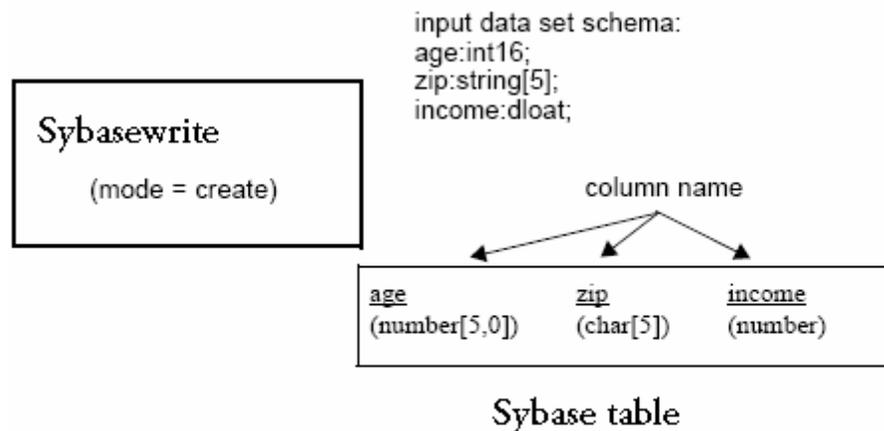
Here is the osh syntax for this example:

```
$ osh " ... op1 | sybasewrite -tablename 'table_2'
-db_name datasource name -user = user101 -password passwd
```

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

Example 2: Creating a Sybase Table

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

Figure 5 Creating a Sybase Table

Here is the osh syntax for this operator:

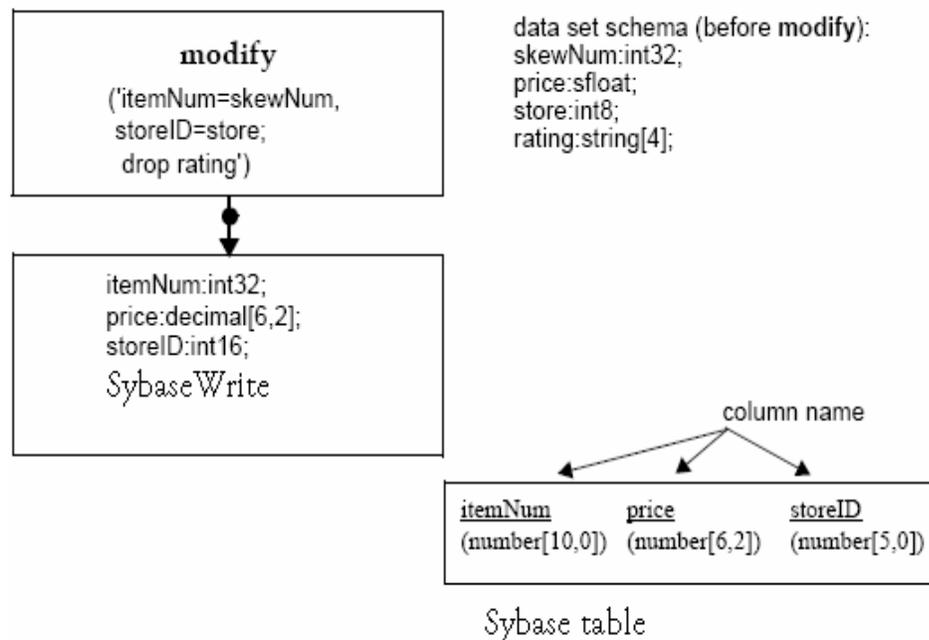
```
$ osh "... sybasewrite -table table_2
-mode create
-dboptions {'user = user101, password = userPword'} ..."
```

The sybasewrite operator creates the table, giving the Sybase columns the same names as the fields of the input DataStage dataset and converting the DataStage data types to Sybase data types.

Example 3: Writing to a Sybase Table Using the modify Operator

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

The next example uses the modify operator:



In this example, you use the modify operator to:

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

Note DataStage performs automatic type conversion of `store`, promoting its `int8` data type in the input dataset to `int16` in the sybasewrite input interface.

Here is the osh syntax for this operator:

```
$ modifySpec="itemNum = skewNum, storeID = store;drop rating"
$ osh "... op1 | modify '$modifySpec'
```

```
| sybasewrite -table table_2
-dboptions {'user = user101, password =
UserPword'}"
```

The sybaseupsert Operator

The sybaseupsert operator inserts and updates Sybase table records with data contained in a DataStage dataset. You provide the insert and update SQL statements using the -insert and -update options.

This operator receives a single dataset as input and write its output to a Sybase table. You can request an optional output dataset that contains the records that fail to be inserted or updated.

Data Flow Diagram

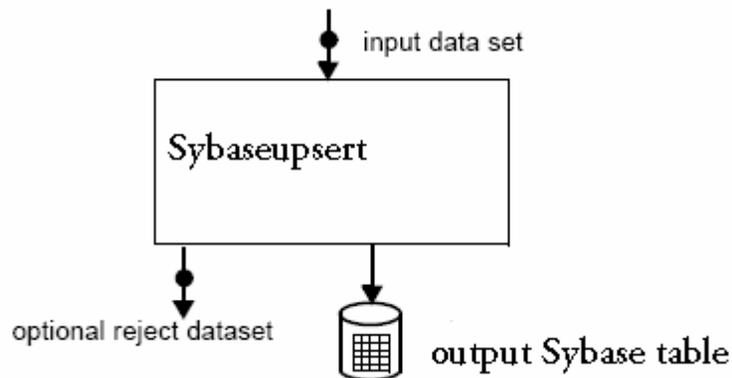


Table 9 sybasewrite Properties

Property	Value
Number of input datasets	1
Number of output datasets 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 dataset when you select the -reject option
Execution mode	Parallel by default, or sequential

Table 9 sybasewrite Properties

Property	Value
Partitioning method	Same You can override this partitioning method; however, a partitioning method of entire cannot be used.
Collection method	Any
Combinable operator	Yes

Operator Action

Here are the main characteristics of sybaseupsert

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.

You 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
```

Your record fields can be variable-length strings. You can 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, by default is, 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 sybaseupsert operator converts all values to strings before passing them to Sybase. The following DataStage data types 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, sybaseupsert produces no output dataset. By using the `-reject` option, you can specify an optional output dataset containing the records that fail to be inserted or updated. Its syntax is:

```
-reject filename
```

Syntax and Options

The syntax for sybaseupsert is shown below. Option values you supply are shown in italic typeface. When your value contains a space or a tab character, you must enclose it in single quotes.

```
sybaseupsert
db_name dsn -user username -password password
-update update_statement
[-insert insert_statement]
[-insertArraySize n]
[-reject]
```

Exactly one occurrence of the `-update` is required. All other options are optional.

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

Sybaseupsert Options

Options for sybaseupsert are as follows:

Table 10 sybaseupsert options

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

Table 10 sybaseupsert options

Options	value
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 dataset. You must designate an output dataset 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 a 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 a 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.

Table 10 sybaseupsert options

Options	value
-insertarraysize n	Optionally specify the size of the insert/update array. The default size is 2000 records.

Example

This example updates a Sybase 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 is the state of the Sybase 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 11 Sybase table

Table before dataflow		Input file contents		DataStage action
Acct_id	acct_balance	873092	67.23	Update
073587	45.64	865544	8569.23	Insert
873092	2001.89	566678	2008.56	Update
675066	3523.62	678888	7888.23	Insert
566678	89.72	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 |
      sybaseupsert -db_name 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 12 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 sybaselookup Operator

With the Sybaselookup, you can perform a join between one or more Sybase tables and a DataStage dataset. The resulting output data is a DataStage dataset containing DataStage and Sybase data.

You perform this join by specifying either a SQL SELECT statement, or by specifying one or more Sybase 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 dataset you are matching is much smaller than the Sybase table. If you expect to match 90% of your data, using the Sybasereade and lookup operators is probably more efficient.

Because Sybaselookup can do lookups against more than one Sybase table, it is useful for joining multiple Sybase tables in one query.

The `-statement` option command corresponds to a 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 Sybase, and outputs a combination of Sybase and DataStage data.

Alternatively, you can use the `-key/-table` options interface to specify one or more key fields and one or more Sybase 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
```

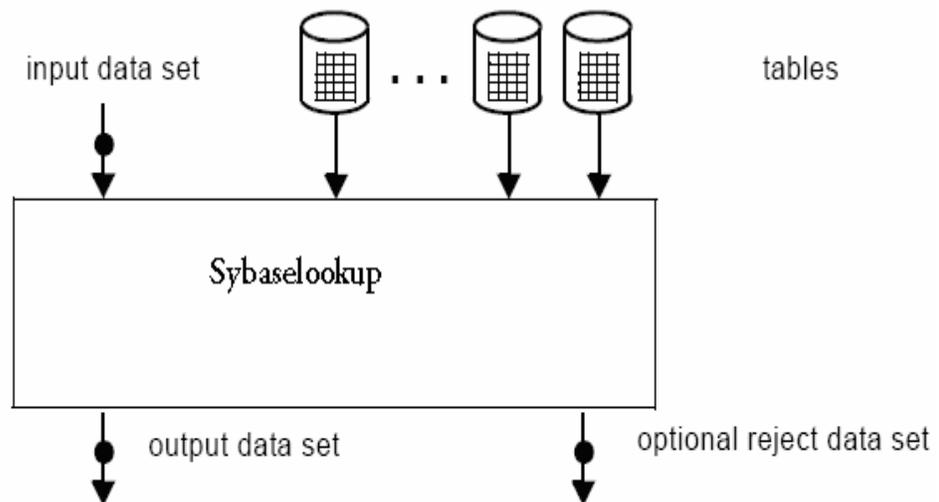
The resulting DataStage output dataset includes the DataStage records and the corresponding rows from each referenced Sybase table. When a Sybase table has a column name that is the same as a DataStage dataset field name, the Sybase column is renamed using the following syntax:

```
APT_integer_fieldname
```

An example is APT_0_Iname. The *integer* component is incremented when duplicate names are encountered in additional tables.

Note If the Sybase table is not indexed on the lookup keys, the performance of this operator is likely to be poor.

Data Flow Diagram



Properties Table

Table 13 sybaselookup Properties

Property	Value
Number of input datasets	1
Number of output datasets	1; 2 if you include the <code>-ifNotFound reject</code> 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)
Preserve-partitioning flag in output dataset	Clear
Composite operator	no

Syntax and Options

The syntax for the sybaselookup operator is given below. Option values you supply are shown in italic typeface. When your value contains a space or a tab character, you must enclose it in single quotes.

```
sybaselookup
  -db_name database -user username -password passwd
  -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.

Sybaselookup operator Options

The sybaselookup operator is a parallel Stage by default. The options are as follows:

Table 14 sybaselookup Options

Options	Value
<code>-db_name</code> <i>database name</i>	Specify the data source to be used for all database connections. This option is mandatory.

Table 14 sybaselookup Options

Options	Value
-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
-table 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 dataset. 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 a SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.

Table 14 sybaselookup Options

Options	Value
-close close_command	Optionally specify a SQL statement to be executed after the insertarray 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 dataset and a table called target, on the key fields lname, fname, and DOB. You can configure sybaselookup in either of two ways to accomplish this.

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

```
$ osh " sybaselookup - }
    -key lname -key fname -key DOB
    < data1.ds > data2.ds "
```

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

```
$ osh " sybaselookup
    -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 Sybase table.

If a column name in the Sybase table has the same name as an DataStage output dataset schema fieldname, the printed output shows the column in the Sybase table renamed using this format:

`APT_integer_fieldname`

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

