SYBASE®

Users Guide

Sybase ETL

4.9

DOCUMENT ID: DC00608-01-0490-01

LAST REVISED: September 2009

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This guide is for users of Sybase® ETL Development.

How to use this book

This book contains these chapters:

- Chapter 1, “Sybase ETL,” is an overview of the Sybase ETL architecture and the feature set of Sybase ETL Development and Sybase ETL Server.
- Chapter 2, “Getting Started,” describes how to get started using Sybase ETL. It familiarizes you with the Sybase ETL Development interface and describes the functions you can perform using the interface.
- Chapter 3, “Projects and Jobs,” tells you how to create, simulate, and execute projects and jobs. It discusses how to use the simulation mode, and how to use templates to create projects and jobs.
- Chapter 4, “Advanced Concepts and Tools,” describes the built-in tools that simplify design work.
- Chapter 5, “Components,” describes the Sybase ETL components that are used to create projects and jobs.
- Chapter 6, “Sybase ETL Server,” provides information on how to use Sybase ETL Server.
- Appendix A, “Function Reference,” describes the built-in functions available in Sybase ETL.
- Appendix B, “Connection Parameters,” describes database configuration options, and provides additional information for some of the supported interfaces.
- Appendix C, “Using ETL for Slowly Changing Dimensions,” describes slowly changing dimensions (SCDs), including some common SCD scenarios, and explains how to implement these scenarios using Sybase ETL.
- Appendix D, “Best Practices,” describes recommendations and guidelines for working with Sybase ETL.

Related documents

See the following documents for more information:

- *Sybase ETL New Features Guide* – describes the new features in Sybase ETL 4.9
- *Sybase ETL Release Bulletin* – contains last-minute information that was too late to be included in the books.
- *Sybase ETL Installation Guide* – describes installation procedure for Sybase ETL.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click Certification Report.

3 In the Certification Report filter, select a product, platform, and timeframe, and then click Go.

4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product, or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The syntax conventions used in this manual are:

Key	Definition
commands and methods	Command names, command option names, utility names, utility flags, Java methods/classes/packages, and other keywords are in lowercase Arial font.
variable	Italic font indicates: • Program variables, such as *myServer* • Parts of input text that must be replaced; for example: *Server*.log • File names
File | Save	Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File | Save indicates "select Save from the File menu."
`package 1`	Monospace font indicates: • Information that you enter in a GUI interface, a command line, or as program text • Sample program fragments • Sample output fragments

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase ETL and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Sybase ETL

Sybase ETL lets you extract data from multiple heterogeneous data sources and load it into one or more data targets using a comprehensive set of transformation functions. It provides a scalable grid architecture that enables parallel transformation processing across operating system boundaries and computers.

Sybase ETL capabilities include:

- Data extraction – extract data from various data sources.
- Data transformation – convert, clean, merge, and split data streams.
- Data loading – load data into a target database using update, insert, delete, or bulk copy statements.

Sybase ETL architecture

Sybase ETL includes Sybase ETL Development and Sybase ETL Server.

Sybase ETL Development, which is available only on Windows, is a graphical user interface (GUI) tool for creating and designing data transformation projects and jobs. This tool provides a complete simulation and debugging environment, designed to speed the development of ETL transformation flows.

Sybase ETL Server is a scalable and distributed grid engine, which connects to data sources, and extracts and loads data to data targets using transformation flows, which are designed using Sybase ETL Development. See “Sybase ETL Server” on page 217.

Sybase ETL Development includes an ETL Development Server that controls the actual processing, such as connecting to databases and executing procedures. To perform parallel execution of jobs and projects, you can add multiple ETL servers on different operating systems within your network. Each server exposes certain services to all other peer servers. Sybase ETL uses the various servers on a grid for parallel execution of projects and jobs, which improves scalability of transformation speed.

Sybase ETL Server connects to the destination or the source database using methods or drivers called interfaces. One of the supported interfaces, the Sybase SQL Anywhere® 11 ODBC driver, which is used to connect to Sybase IQ and Sybase SQL Anywhere, is automatically installed by the Sybase ETL Development installer. To install the other supported interfaces, see the respective vendor documentation.

Note Sybase SQL Anywhere 11 is packaged with Sybase ETL Server and must be installed manually.

You can also use the command line to perform jobs and projects execution on all supported platforms. See “Using ETL Server to execute projects and jobs” on page 221.

Note To perform parallel execution of projects and jobs using Sybase ETL Development, install the Sybase ETL Server, which is available as a separate executable. Perform parallel execution of projects and jobs only if you are running more than one ETL Server.

Registering ETL servers

All the ETL servers that you add to the grid must be registered. You can use the Engine Manager, available within Sybase ETL Development to register ETL servers. See “Using multiple engines to reduce job execution time” on page 76.

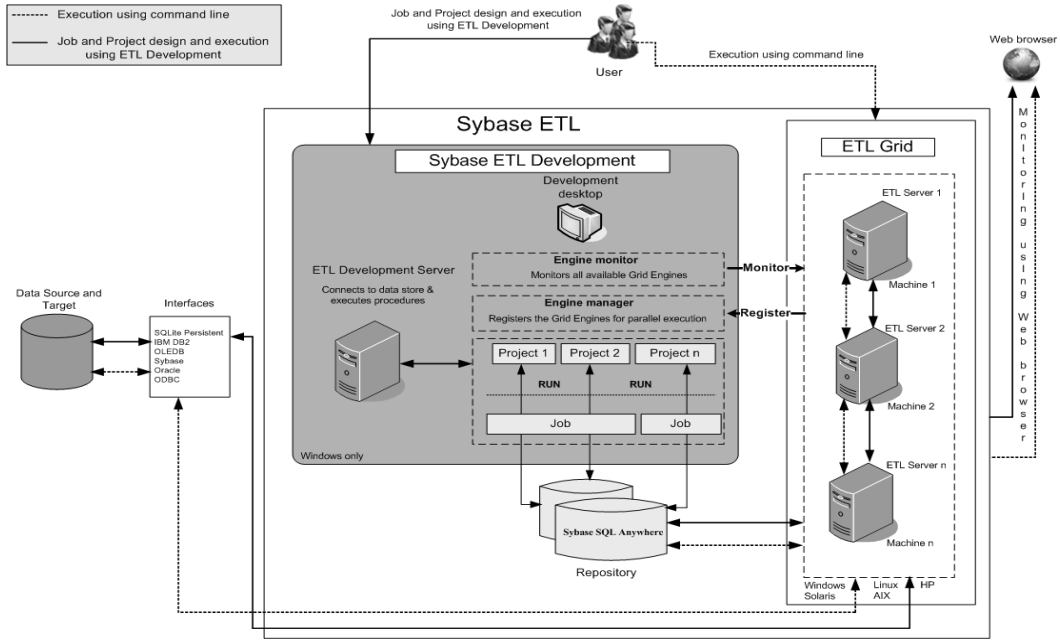
Monitoring ETL servers

You can use the Engine Monitor, which is also available from Sybase ETL Development, to monitor the servers in your network. See “Engine Monitor” on page 78. You can also use a Web browser to monitor remote projects and jobs that are started from a command line. See “Monitoring projects and jobs using a Web browser” on page 226.

Note The terms grid engine and ETL server are used interchangeably in this guide.

Figure 1-1 on page 3 provides a graphical representation of the Sybase ETL architecture.

Figure 1-1: Sybase ETL architecture



Sybase ETL concepts

This section introduces Sybase ETL concepts.

Repositories

Repositories contain all data and information related to Sybase ETL objects, projects, and jobs.

During a session, multiple repositories are simultaneously accessible. You can copy and transfer projects between repositories, which allows you to separate your production repository from your development repository. See “Managing projects” on page 25.

A repository usually belongs to a single client, such as a department or firm. More than one client can use the same repository. Each client can support any number of users; each user has a user name and password that controls access to information.

Note Do not manually manipulate data in the repository tables. Sybase cannot guarantee the functionality of a repository after it has been manually manipulated. It can also make the repository unusable, and your work may be lost.

Projects and jobs

A project is a collection of components, links, and transformation rules. Each project contains one or more steps that are simulated or executed sequentially when the project is run. Components connect to various data sources from where they read data to transform based on the transformation rules. A project consists of various components that can be freely arranged.

Within a job, you can run multiple projects sequentially or in parallel. Jobs control the order in which projects are executed. Jobs can be scheduled and monitored.

Running projects and jobs

You can run projects using either simulation or execution mode.

Both modes perform all functions of the components included in a project, including the physical transfer of data into the data targets, which are also called data sinks.

Simulation mode enables you to:

- Run projects with unsaved changes.
- Monitor and validate the transformation process step by step. Data flow is visible on any link and within any component included. You can also inspect any component, and modify mappings and calculations.

  After making changes, you can reinitialize the component with the new settings and step to the next component. You do not have to restart the simulation.

See “Simulating projects interactively” on page 28.

Note If you are running a project in simulation mode and your objective is to test transformation rules, you may want to use a test data target.

Execution mode lets you:

- Run projects and jobs that have been saved to the repository. Unsaved changes are not executed.
- Execute the project and reflect the changes in the data sink. You cannot monitor the transformation process step by step.

Note You can execute projects and jobs either from Sybase ETL Development, or as a scheduled task. See “Managing jobs and scheduled tasks” on page 57.

Customizing a project

You can create data transformation projects without using programming code or SQL statements. For example, you can:

- Use the Query Designer to generate select statements inside queries, lookup definitions, preprocessing and postprocessing SQL.
- Use the data mapping features of the links between the components to map attributes between data sources and data sinks.
- Use the built-in Create Table From Port command of the component you are using, to create temporary or persistent staging tables, or to create tables in the destination database.
- Use the Content Explorer to browse schema information and data content of all connected data sources.
- Use the XML via SQL Data Provider component to read hierarchical XML documents and automatically generate a relational structure.
- Schedule execution of projects and create jobs within Sybase ETL Development.

Additionally, when you have complex data transformation requirements, you can use:

- Manually optimized SQL select statements to adjust the data extraction process.

- SQL statements to apply data manipulation commands in different phases of the transformation.
- JavaScript to write procedures, perform complex calculations, or manipulate objects in the operating system environment.
- Indirection via expressions to dynamically control your projects by using environment or user variables.

Components

Stepping a component record-by-record

In simulation mode, many transformation components allow you to step through the current set of data and immediately visualize the result of any applied transformation.

Adaptable port structure and mapping

All data within a project flows through component ports called IN-ports and OUT-ports. Each port owns the structure of the data flow. You can change the port structures of all components for which the structure is not directly dependent on component configuration. Attributes added to the port structure can be referenced immediately inside the component.

When connecting components, Sybase ETL attempts to create a standard mapping between the OUT-port and the IN-port. You can modify the mapping on a connection in the Mapping window. To open the Mapping window, right-click the connecting link, and select Mapping. See “Viewing current mappings” on page 30.

SQL statements

Most of the data delivered by Data Providers is defined by using SQL statements stored in the Query property. Sybase ETL supports a modified set of the SQL92 standard.

You can manually write or copy SQL statements from existing projects into the Query property. If you do not want to work with the details of SQL92, use the Query Designer to draw the query and automatically generate the SQL statement.

JavaScript

ETL supports the JavaScript language for expressions and procedures used in components, to transform and manipulate data within the transformation process.

JavaScript expressions enclosed in square brackets can also be used within component property values allowing a parameterized configuration. Square Bracket Notation (SBN) is a widely applicable indirection mechanism within the Sybase ETL environment. You can apply Square Bracket Notation, for example, within SQL statements and file name specifications to compute and assign values dynamically at runtime.

Datatypes and data formats

Data source datatypes are preserved during transformation.

Internally, Sybase ETL distinguishes between string and numeric datatypes. The Standardize Data Format option of the data providers or data sinks automatically converts data to a standard format, which is then converted to a format, which the target database can process. You need not manually convert various date and number formats when working with different databases.

By default, the Standardize Data Format option is selected. However, if you experience problems with date or number fields, you can disable this option and manually convert the data. See “Customizing preferences” on page 19.

Unicode support

All components are designed to process and support Unicode and multibyte data. You can use Unicode-enabled transformation functions in calculations, scripts, and procedures. The level of Unicode support for Sybase ETL allows you to:

- Extract, transform, and load data containing Unicode characters
- Use Unicode characters in component properties:
  - File or directory names
  - Metadata, such as, table or attribute names
  - Connection settings, such as, database, schema, user, or password
  - Transformation rules

Tools

Structural and catalog information from all connected data sources is accessible through Sybase ETL tools. You can browse through schema information or data, and even create new database objects using these tools. See “Advanced Concepts and Tools” on page 51.

CHAPTER 2

Getting Started

Starting Sybase ETL

1 In Windows, select Start | Programs | Sybase | Sybase ETL Development 4.9 | Sybase ETL Development.

The login window displays:

- Connection – `Repository`
- Client – `transformer`
- Client user name – `TRANSFORMER`
- Password – `transformer`

These values are automatically set the first time you log in. On subsequent logins, you might need to select or enter this information.

Click Logon

2 In the Navigator, click Repository | *TRANSFORMER.transformer.Repository* | Projects to open the list of available projects.

Note The project list displays the demonstration projects packaged with the product. Every demonstration project contains an example of how to use a component or implement a scenario.

3 Double-click an existing project name to open it, or to create a new project, right-click Projects and select New.

Setting up a new user account on the demo repository

1 From the Sybase ETL Development interface, select File | Open Repository.

2 Enter a new Client User name.

Note Do not change the client name if you want to access the demo projects and jobs.

3 Enter a password.

4 Select Register new user.

5 Select Show all objects. If you do not select this option, you cannot access the demo projects and jobs.

6 Click Logon.

7 Reenter the password and click OK.

Working with the Sybase ETL Development interface

The Sybase ETL Development interface consists of:

- Design window – used to create projects and jobs based on transformation rules.

- Component Store – used to find components for your projects.
- Navigator – used to locate projects, jobs, and templates. Also displays recently accessed projects, jobs, and templates.
- Assistant window – used to help with current tasks.
- Properties window – used to set the properties of components.

Navigator

Use the Navigator to:

- Administer the repository
- Navigate and browse the repository
- Administer projects and jobs
- Execute projects and jobs
- Administer user accounts

Administering the repository

A Sybase ETL repository is a collection of tables that saves and maintains all data related to projects, jobs, and session parameters. You can use Sybase SQL Anywhere databases as the Sybase ETL repository.

Note Do not manually manipulate data in repository tables; doing so can make the repository unusable, and you may also lose data. Sybase cannot guarantee the functionality of a repository after it has been manually manipulated.

To access projects or jobs, log on to the respective repository. To open a repository, you must assign at least one client and one client user. A client can have multiple users.

❖ **Opening a repository**

1. Select File | Open Repository. Or, in the Navigator, right-click Repositories and select Open Repository.
2. Select a repository from the Connection list and click Logon.

❖ **Closing a repository connection**

1. In the Navigator, right-click the repository name and select Close Connection.
2. Click Yes to confirm that you want to close the connection and all open projects and jobs.

Closing a repository ends all user sessions currently connected to the repository.

❖ **Closing a client user session**

1. Right-click the repository name in the Navigator and select Close Client.
2. Click Yes to confirm that you want to close the client and all open projects and jobs.

❖ **Adding a repository**

1. Select File | Open Repository to open the Repository Logon window.
2. Click Add.
3. Enter the parameters for your new repository connection and click Save.

To access the new repository, you must create at least one client and one client user definition.

❖ **Creating a client and a client user**

1 In the Repository Logon window, enter a client name in the Client field.

2 Enter a Client User name. The name must be alphanumeric, can contain up to 255 characters, and cannot start with a number.

3 Enter a password.

4 Select Register New User.

5 If the client user is entitled to see all existing projects within the client, select Show All Objects.

6 Click Logon.

7 Reenter the password and click OK.

Note You can also create a user from the Use Accounts window. See "Creating a user" on page 14.

❖ **Editing a repository**

1 Select File | Open Repository.

2 Select the repository you want to modify and click Edit.

3 Make changes and click Save.

❖ **Removing a repository**

1 Select the repository from the Connection list and click Remove.

2 Click Yes to confirm the removal.

Navigating and browsing a repository

In the Navigator, the hierarchical tree list represents:

- Open repositories
- Client user sessions to the open repositories
- Objects stored in the repository, such as projects, jobs, and templates
- Recently opened projects, jobs and templates

A repository can be simultaneously opened by multiple client user sessions. A client user is part of a client. Both client users and clients are registered when they log on to the repository.

The following example shows the tree structure:

```
Repositories
-- <RepositoryName1>
---- <ClientUser1>.<Client1>.<Repository Name1>
------ Recent
-------- Recently opened projects
-------- Recently opened jobs
-------- Recently opened templates
------ Projects
-------- Project_1
-------- Project_2
-------- Project_N
------ Jobs
-------- Job_1
-------- Job_2
-------- Job_M
------ Templates
-------- Template_1
-------- Template_L
---- <ClientUser1>.<ClientM>.<Repository Name1>
---- <ClientUserN>.<Client1>.<Repository Name1>
-- <RepositoryName2>
```

Administering projects and jobs

From the Navigator, you can administer projects and jobs. See Chapter 3, “Projects and Jobs.”

Administering user accounts

From the Navigator, you can:

- Create a user
- Remove a user
- Change password
- Change visibility

Only registered client users can access a repository. You can register a client user in the Repository Logon window or in the User Accounts window.

❖ **Creating a user**

1 In the Navigator, right-click a repository name and select User Accounts.

2 Click Add User.

3 Enter the user name. The user name must:

- Contain only alphanumeric characters
- Start with an alphabetic character
- Contain up to 255 characters
- Not be empty

4 Enter a password.

5 Reenter the password.

6 Select Show All Objects to show objects belonging to other repository users.

7 Click OK.

❖ **Removing a user**

1 In the Navigator, right-click a repository name and select User Accounts.

2 Select the user you want to remove and click Remove User.

If the user is connected to the repository, confirm that you want to remove the user and close all open projects and jobs. Click Yes.

3 Enter the password of the user you are removing and click OK.

❖ **Changing passwords**

1 In the Navigator, right-click a repository name, and select User Accounts.

2 Select the user for whom you are changing the password.

3 Click Change Password.

4 Enter the existing password of the user and the new password. Reenter the new password.

5 Click OK.

Assistant window

The assistant window assists you with your current task.

Properties window

Use the Properties window to:

- View and edit the component properties
- Identify mandatory component properties
- Allow dynamic evaluation of component properties
- Encrypt component properties
- Add custom properties to a component and edit their values
- Access the component configuration window
- Enable transactionality for a project or job

See Chapter 5, “Components,” for component specific property settings.

Viewing and editing component properties

To view and edit component properties and values, select the component in the Design window. All properties of the selected component appear in the Properties window.

Identifying mandatory properties

A property name displayed in **bold** text in the Properties window indicates that the property is required for the component to operate correctly. All other properties are optional; use them to fine-tune and configure the component.

Allowing dynamic values

Select the Evaluate option to allow expressions within component property values. The Evaluate option lets you compute and evaluate dynamic property settings at runtime instead of assigning static values at design time.

For some property items, the Evaluate option is by default, selected.

Use the Eval checkbox in the Properties window to enable or disable evaluation for a property. Once the Evaluate option is activated you can include JavaScript expressions in the corresponding field using square bracket notation (SBN). See “Evaluating SBN expressions” on page 88.

Encrypting properties

Project and job data, as well as property values, are stored in the Sybase ETL repository. Most of the records in the Sybase ETL repository are not encrypted. For the password property, the Encrypt option by default, is selected.

To encrypt property values, right-click a property name in the Properties window, and select Encrypt.

Alternatively, select the Encrypt checkbox next to the property value.

Adding and editing custom properties

Use the Properties window to add or edit custom component properties. Like other properties, custom properties also incorporate a variable that can be referenced in expressions or user-defined procedures.

See “Custom properties” on page 89.

Accessing the component configuration window

In the Properties window, click the Property icon to open the configuration window for the selected component.

Note Some components do not have a configuration window.

Enabling transactionality for projects and jobs

In the Properties window, select Propagate Rollback to enable transactionality support for the generated jobs or projects. When you select this option, data is committed at the end of the write operation for a successful execution, and rolled back for an unsuccessful execution. If you do not select the option, the project does not enforce a transaction rollback on successful components, if one or more components fail. Failed components rollback their own transactions. Jobs also, when Propagate Rollback is not selected, do not enforce transaction rollbacks on successful projects, if one or more projects fail. The failed projects roll back their own transactions.

Note If Propagate Rollback is selected for a project or job, and there are multiple components that are a target for the same table, also select the Shared connection property for all these components. Otherwise, ETL may stop responding.

Design window

Use the Design window to:

- Create and modify projects and jobs. See “Projects and Jobs” on page 25.
- Simulate and execute projects. See “Simulating a project” on page 27 and “Executing a project” on page 35.
- Execute jobs. See “Managing jobs” on page 36.

Adding components to the Design window

To create a project or job, add and connect components, and set their properties. You can add components to projects and jobs using any of these procedures:

- Select the component in the Component Store and drag it to the Design window.
- Double-click a component in the Component Store.
- Right-click a component in the Component Store and select Add.

In addition, components can also be added to projects using this procedure:

- Right-click the port of an existing component in the Design window and select Add Component. Point to the component type and select the component you want to add. The component is added before or after the existing component, depending on whether the selected port is an IN-port or an Out-port.

Deleting components from the Design window

1 In the Design window, select the component to delete.

2 Right-click and select Delete.

Processing general commands

1 Right-click anywhere in the Design window to open the general project menu. This menu displays general commands, such as Close and Print.

2 To perform an action on the component, right-click the component and select the action you want to perform.

Component Store

The Component Store consists of several sections that group components by general purpose.

To add components from the Component Store to a project or job:

- Drag the component onto the Design window
- Right-click a component and select Add
- Double-click a component

Customizing preferences

Use the Preferences window to customize groups of settings in Sybase ETL Development:

- Workbench
  - Appearance

- Data Viewer
- Query Designer

- Engine
- Performance Logging

❖ **Customizing preferences**

1 From the main Sybase ETL Development interface, select File | Preferences.

2 Select Appearance, and set these options:

- Locale for user interface display – select the locale language for your environment: _de (German), _en_US (U.S. English), or _en_GB (UK English). The default is _en_US.
- Show assistant for creating projects – view the assistant that guides you through completing a project, and displays information on the current state of the open project.
- Default font for displaying text – select the font for displaying text file contents in the Text Data Provider and Text Data Sink component windows. This setting is useful when you work with non-Western character sets, such as Unicode. The default font is Monospaced. The recommended fonts for displaying text are Dialog or Monospaced.
- Default font for displaying data – select the font for displaying port data throughout the application. This setting is useful when you work with non-Western character sets, such as Unicode. If you are installing Sybase ETL Development for the first time, the default font is Dialog. If you have previously installed this version of Sybase ETL Development, the font is set to the previously defined value. Sybase recommends that you set the font to Dialog.

  Note To use "Default font for displaying text" and "Default font for displaying data" options, Sybase recommends that you install the files for East Asian languages. In the Windows Control Panel, click the Regional and Language Options, select the Languages tab, then select "Install files for East Asian languages".
  Enabling this option installs the required fonts for displaying Unicode characters.

- Create a new project on startup – to automatically create a new project when you start Sybase ETL.

- Create links automatically when components are added – to automatically create links between an existing component and new components added to the project.
- Default action on double-clicking a connection – specify whether to open the Mapping window (default) or the Preview window when you double-click a connection during simulation. The Mapping window displays the mapping information and the Preview window provides a preview of the connection data.
- Display qualified transformation objects – to prefix the owner name to the name of the objects in the Navigator. For example, if this option is selected, a project name appears in the Navigator as:

  ```
  TRANSFORMER.Demo Character Mapper
  ```

  where TRANSFORMER is the name of the client user who created the project.
- Use unique object names – to enforce unique project and job names on a repository connection.
- Show password in component tooltips – displays, in component tooltips, the password used to log in to the underlying database. By default, the password appears in the tooltips as a series of asterisks.
- Use enhanced color accessibility – to change the color of the component ports to enhance support for users with color disabilities. Selecting this option changes the default color of the port from yellow to blue, enabling users with color disabilities to distinguish between port states.
- Use vertical component layout – to display the alignment of projects and jobs vertically from top to bottom instead of the left to right, which is the default.
- Show information dialogs – to perform actions without being interrupted by information prompts, unselect this option.
- Number of recently opened projects, jobs, and templates to display – specify the number of recently accessed projects, jobs, and templates you want to view in the Navigator and the File menu.
- Open the last repository automatically – to connect automatically to the last logged on repository, on restart.

- Save repository client password – save the client password after you log in to the repository. If selected, you do not have to provide the password during your next login, as is automatically provided in the Password field of the Repository window.

3 Select Data Viewer and specify the maximum length of an exported data field. The default value is 255. Data fields longer than the value you specify here are truncated in the export files.

4 Select Query Designer and set these options:

- Enable delete functionality of database objects – delete all records of a selected table when you right-click the table in the Query Designer and select Truncate Object.
- Default number of records to retrieve from the Query Designer – specify the default number of data records to be retrieved by the Query Designer. The default is 25.
- Create joins automatically – to automatically create joins between identical attribute names used within tables or views.
- Use brackets when creating joins – to automatically surround join clauses with brackets in the generated query. For example, the select statement would appear as:

```
select * FROM SALES ((<join statement 1>)
<join statement 2>)
```

- Default number of recently accessed tables and views to display – specify the number of recently accessed tables and views you want the Query Designer to display in the Recent tab. The default value is 25.

5 Select Engine and set these options:

- Start local engine during application startup – to start the local engine when you start Sybase ETL.
- Interval between engine monitor updates (sec)– specify the number of seconds to wait between updates of the Engine Monitor. The default is 5 seconds.
- Rate of simulation (msec) – control the simulation rate by setting simulation trace delay. The simulation trace delay option accepts values between 10 and 9999 milliseconds. The default is 250 milliseconds.

- Grid engine ping timeout (sec) – specify the number of seconds allowed for accessing the grid engine before starting or restarting a local grid engine. The default is 60 seconds.
- Interval between progress monitor updates (sec) – specify the number of seconds to wait between updates of the progress monitor for a job execution. The default is 5 seconds.
- Allow selection of the execution engine – to specify the grid engine to be used for project execution.
- Execution engine server – specify the IP address of the primary grid engine server.
- Execution engine port – specify the port address of the primary grid engine server.

6 Select Performance Logging and specify the detail level for logging performance data:

- 0 – performance data is not written to the repository.
- 1 – performance data is written to the repository. This is the default value.

7 Click Save. For some of the changes to take effect, you may be prompted to restart Sybase ETL. If you select not to restart when prompted, the changes take effect the next time you start Sybase ETL.

Troubleshooting

The Sybase ETL installer creates an initial set of data sources. If these repository data sources are lost for any reason, Sybase ETL cannot open until you restore them. To restore the initial set of ODBC data sources of the demo repository:

1 Configure the ODBC user data source:

a Select Start | Settings | Control Panel | Administrative Tools | Data Sources (ODBC).

b Click Add.

c Select SQL Anywhere 11. Click Finish.

d Enter “DEMO_Repository” as the ODBC data source name.

e Click the Login tab and enter “dba” as the user ID and “sql” as the password.

f Click the Database tab and in the Start line field, enter “C:\Program Files\Sybase\ETLDevelop49\dbeng11.exe.” This is the default installation location.

g In the Database file field, enter “C:\Program Files\Sybase\ETLDevelop49\Demodata\demo_rep.db.”

h Return to the ODBC tab and click Test Connection to verify the connection.

2 Set up the repository connection in the Repository Logon window:

a Select File | Open Repository.

b Select Repository from the Connection list and choose:

- Edit, or,
- Add, and enter a name for the connection.

c Select ODBC from the Interface list.

d Select DEMO_Repository from the Host list.

e Click Save.

3 Configure the additional ODBC user data sources required by the projects in the demo repository:

- Driver – SQL Anywhere 11
- Name – ETLDEMO_DWH; database – demo_dwh.db
- Name – ETLDEMO_GER; database – demo_ger.db
- Name – ETLDEMO_US; database – demo_us.db

The database files for these user data sources are also located in the *Demodata* folder of the installation directory.

CHAPTER 3

Projects and Jobs

Managing projects

Projects consist of components and links, which connect components through their ports. There are basic operations that involve projects, such as creating, deleting, renaming, and saving, and there are complex operations, such as simulation.

A Sybase ETL project starts with one or multiple source components and ends with one or more destination components.

These are the Sybase ETL components:

- A Data Provider component is usually connected to a transformation component, a processing component, or a data sink component.
- Transformation components and processing components have IN-ports and OUT-ports, and can have adjacent components of any other type.
- If a transformation component allows multiple input data streams, multiple originating source components are required.
- If a transformation component has multiple outputs of data streams, you can connect each data stream with a component.

❖ Creating a project

1 In the Navigator, right-click Projects and select New | Project. Alternatively, select File | New | Project.

2 Drag the components for the project from the Component Store onto the Design window.

❖ **Modifying a project**

1 In the Navigator, double-click the project you want to modify.

2 Make the changes and save the project.

❖ **Unlocking a project**

A project locked by another user client opens in read-only mode.

- To make the project available for reading or writing, click Unlock and Open on the window that appears when you open a locked project.

❖ **Copying a project**

1 Double-click the project you want to copy to open it in the Design window.

2 In the Navigator, right-click the project and select Save As. Alternatively, select File | Save As.

If you are working with multiple repositories, select the target repository.

3 Enter a name for the new project. A copy of an existing project is created, leaving the original untouched, and storing no references to the originating project.

❖ **Transferring a project**

If you are using multiple repositories, you can copy complete projects from one repository to another, maintaining references to the originating project. For example, you may want to move a project from a development repository to a test or production repository. By storing the references to its origin, the transfer recognizes the project the next time it is initiated and selectively replaces everything related to the incoming object.

1 In the Navigator, right-click the project you want to transfer and select Transfer. Alternatively, select File | Transfer.

2 Select the repository where you want the project to be transferred.

Note The transfer option copies project definitions and execution properties. Related data, such as, parameter sets are not transferred.

❖ **Deleting a project**

1 In the Navigator, right-click the project and select Delete.

2 Click Yes to confirm the deletion.

❖ **Renaming a project**

1 In the Navigator, right-click the project and select Rename.

2 Enter a new name for the project and click OK.

❖ **Resetting execution properties**

To reset loading options for incremental load:

1 In the Navigator, right-click the project, and select Reset Execution Properties.

2 Click Yes to confirm that you want to reset execution properties. The current value of the Load Index Value (DB Index Load component) is reset.

Simulating a project

Simulating a project lets you monitor and validate your transformation process step by step. In contrast to executing a project, simulation allows you to:

- Run projects that have unsaved changes.
- View the data at any stage of the transformation process.

During the final steps of a simulation, data is either written into the data sinks, or rolled back, depending on your selection. Many transformation components, such as the Data Calculator, allow you to change transformation rules and sample values during simulation, to validate your rule base for all potential content.

Note You can simulate a project only after all components have been properly initialized.

The basic functions of a simulation includes:

- Starting a simulation
- Stepping through a single component
- Tracing multiple components at a predefined pace

- Viewing the data flow on the connecting link or within the component
- Modifying and reinitialize the component to continue to simulate the data flow
- Committing or rolling back the data

At a more detailed level, you can:

- View data content on connecting links
- View input and output data inside a component
- Modify properties or calculations, so you can change transformation rules and sample values to validate your rule base
- Step through a component again after modifying a calculation or property
- Perform what-if scenarios
- Take multiple steps through the project

Simulating projects interactively

To run the project interactively, select Run | Trace. You can stop the simulation at any point, and select Run | Step or Run | Step Through to manually step through the remaining project components.

❖ **Simulating projects interactively**

1 To start a simulation, click Start on the toolbar:

   - All components of the project are initialized.
   - All connections within the project are validated.
   - All pre-SQL statements in the projects are executed.
   - Data for all static lookup components is retrieved and cached. Any change of data in lookup tables during simulation is not reflected in the simulation process.

2 Select a component and click the Step icon on the toolbar to execute the component.

   "Stepping a component" means executing or processing a single component. The data records that are processed during a single step are the records currently populating the IN-ports of the component.

If a component is stepped multiple times and no other components are stepped in between, the number of records received or forwarded remains constant. Many components can be stepped from both inside the component and outside in the project view.

3 View the data flow on the connecting link or within the component.

- To view data throughout transformation, examine the link between components or the ports of a component. Other components such as the Data Calculator and the Data Splitter include built-in preview capabilities.
- To view data on the connecting link, right-click and select Preview.
- To view data currently at the port, right-click the port, and select Preview.

Note The Preview option is disabled when there are no processed records or when no simulation data is available.

- To view data from inside the component, double-click the component, or click the Rule icon in the Property window. Use this option to see the impact of transformation rules from within components, such as the Data Calculator or the Data Splitter.

4 Modify and initialize the component.

After you modify a component, you can reinitialize the component to continue simulating the data flow. You need not restart a complete simulation for the current project.

a Double-click the component to modify its properties in the Properties window.

b Save the changes.

c Right-click the component and select Initialize.

5 When all data has been processed, you are prompted to select one of these:

- Execute Post-Processing as for successful execution – to commit all tasks performed by the transactional components and to reset the project to its initial state.
- Execute Post-Processing as for failed execution – to rollback all tasks performed by the transactional components and to reset the project to its initial state.

Click Yes to confirm resetting of the interactive trace. This clears all port buffers, releases temporary tables, and closes all database connections and temporary files.

If you click No, all open database connections and port buffers are retained. You can inspect, reconfigure, and then restep individual components.

To reset trace to commit or roll back data for the transactional components, click Reset on the toolbar, or select Run | Reset. Click Yes to confirm.

❖ **Setting the trace delay**

You can control the simulation rate by setting the simulation interactive trace delay option.

1 Select File | Preferences | Engine.

2 Modify the value in the Rate of simulation field. You can enter values between 10 and 9999 milliseconds. The default value is 250 milliseconds.

Viewing current mappings

The Mapping Definition window shows the current mapping between attributes of adjacent input structures and output structures.

1 Right-click the connecting link, and select Mapping.

2 In the Mapping Definition window, select:

- Display structure to view all attributes of the connected port and their current mappings.
- Display structure and values to view the fields as well as the values of the current record. This view shows the current content of the port connecting to the link. If the port contains no data, only the port structure is shown in this window. You can populate data in a port by stepping through your project until you reach the port.

Applying automatic mappings

To create mappings, select one of these predefined mapping sequences from the Mapping menu:

- Create mapping by Order – sequentially maps the port attributes of the IN- structures and OUT- structures. If the number of attributes is different on both sides, some of the port attributes are not mapped.

- Create mapping by Name – maps port attributes of the IN- structures and OUT- structures, according to their names.
- Create mapping by Name Case Sensitive – maps the port attributes of the IN- structures and OUT- structures, according to their case sensitive names.
- Create mapping by Prefix – maps the port attributes of the IN- structures and OUT- structures by name, ignoring the specified prefixes.
- Create mapping by Best Match – maps the port attributes of the IN- structures and OUT- structures that sound alike.

Applying manual mappings

To manually create a single mapping, select a connection point and drag it to the connection point of a port attribute.

To change a current mapping, select the mapping line at the connection point and drag it to an unmapped port attribute.

To delete a single mapping, right-click the mapping, and select Delete.

To delete all mappings of a link, select Remove All from the Mapping menu.

Viewing mapped attributes

By default, the Mapping Definition window displays all the port attributes of the IN structures and OUT- structures. To view only the mapped attributes, click the “Display only mapped attributes” icon on the toolbar.

Enabling synchronized attribute scrolling

To enable synchronized scrolling between the attributes of the IN- structures and OUT- structures, click the “Synchronize attribute scrolling” icon on the toolbar.

Managing port attributes

You can add and delete port attributes, or modify the settings or existing attributes in the Structure Viewer window.

❖ **Adding attributes to the port structure**

1 In the Design window, right-click the port, and select Edit Structure.

2 In the Structure Viewer window, select Actions | Add, or right-click the attribute and select Add.

3 Enter a name for the attribute. The names for port attributes must start with an alphabet character and can contain only alphanumeric characters. Names cannot be reserved JavaScript keywords. See “Variables” on page 62.

Select Populate Attribute to add the attribute to multiple port structures. The new attribute is added to all port structures participating in the selected connections and is automatically mapped. Click OK.

4 Specify the other details.

❖ **Deleting attributes from the port structure**

1 In the Design window, right-click the port, and select Edit Structure.

2 In the Structure Viewer window, select Actions | Remove, or right-click the attribute and select Remove.

❖ **Modifying port attributes**

1 In the Design window, right-click the port, and select Edit Structure.

2 In the Structure Viewer window, change the attribute settings, and click Save.

See “Managing port structures” on page 90.

Modifying datatypes

When you modify datatypes of a record structure, you modify the internal logical representation that Sybase ETL uses for the record structure during transformation. This does not change the data structure definition of the source or destination tables. Make sure that the data structure of the final Data Sink is compatible with the content you are generating.

Viewing a simulation flow

After you start simulation, the flow of the simulation is made visible through:

- A green dotted box indicates the active component and moves with each step from one component to the next.
- The number of records appears on the link, which follows the box movement.

The number of records being processed within each single step depends on the current value of Read Block Size of the previous component with a Read Block Size property.

Selecting a small number of records is useful while performing a simulation. A large number for Read Block Size can significantly enhance performance when a project executes.

Stepping from current and selected components

When a simulation starts, the first component to be executed is indicated with a dotted green box.

When you step through the project without making modifications, the box moves from component to component, displaying the success or failure icon until the end of the simulation.

You can select a component other than the current one to inspect or change its properties. The selected component is indicated by a solid green box.

The current component is executed next, when you click Step on the toolbar or select Run | Step. To inspect or change a component that is different from the current component, click it. The solid green box highlights the selected component.

After making changes, resume the simulation from either the selected or the current component:

- To resume simulation from the selected component, right-click and select Step.
- To resume simulation from the current component, click Step on the toolbar.

Note If you right-click an unprocessed component, and select Initialize and Step, the component is initialized and stepped, and the next component to be processed is highlighted.

Forwarding and backward-forwarding components

The visual flow of the simulation as indicated by the box is straightforward in most projects; the box moves from one component to the next. However, the flow of a project simulation does not necessarily progress in only one direction; flow direction depends on the components used within the project.

Forwarding components, such as Data Calculator and Character Mapper, receive a number of records, apply transformation to those records, and forward the records. The number of records processed in a single step is determined exclusively by the number of records received from the preceding component.

Other components override previous Read Block Size settings. The Staging component is designed to work on the entire result set of the data stream, as defined with the query of the Data Source component. The component does not process and forward any data records until the entire result set is delivered to the IN-port. The Staging component uses its own Read Block Size property to resize the number of records forwarded with the next step. See Chapter 5, “Components” for detailed explanation of the behavior of every component during the simulation.

Previewing data from multiple locations

Right-click any connecting link, port, or component, and select Preview to open the Content Browser window that displays the data currently available at the selected location.

Note The Preview option is not available when there are no processed records or when no simulation data is available.

The Content Browser window includes tabs that allow you to simultaneously display multiple previews from multiple locations. You may find it useful to preview the content of both the IN-port and OUT-port of a component.

To save the displayed data to a definition file, click the Export data icon on the toolbar. Specify the options for exporting data.

Partial execution or initialization during simulation

Restarting an entire simulation after making modifications to a single component can be time consuming, especially if your project has a large number of input records in a project that consists of large number of components. To multi-step through a project to your point of interest, select the component and choose Run | Step through or Run | Start.

Simulating up to a certain component

To validate your current project by starting from a component somewhere in the middle of a project, select the component, and then select Run | Start Through. The simulation starts the current project, processes all components between the current and the selected component, and processes the selected component.

Impact of Read/Write Block Size

The number you enter as the Read Block Size defines the number of records fetched by the component during a single simulation step. Set the Write Block Size to define the number of records to be written. Most data provider components possess a Read Block Size property. You can customize the Write Block Size for most of the data sink components. You can customize both reading and writing values for transformation components, such as the Staging component.

Note The Block Size property is evaluated during project simulation as well as project and job execution. A small number might be suitable for simulation purposes, but slows execution. In simulation, the Block Size is restricted to 32K.

Controlling multiple data streams

Most projects consist of a single stream of components connected through links. However, you can also set up a single project that has multiple unconnected data streams. You cannot predict the order in which the streams are processed.

If you use multiple data streams, Sybase recommends that you design a project for each data stream so that all components within a project are connected to each other. This lets you control data streams by connecting projects to form a job process flow.

Executing a project

Execute projects in the default grid engine using one of these ways:

- For projects currently open in the Design window, select Run | Execute, or click the Execute icon on the toolbar.

Execution impacts the state of the simulation. If you try to execute an unsaved project, you are prompted to save the project. If you save the project, all simulation data for the project is lost, and the project is executed.

Note Before execution, you must save project changes as the project definition is read from the repository. If you do not save the changes, execution does not start.

- In the Navigator, select the project you want to execute. Right-click and select Execute Project.

The Execution Monitor displays. See “Execution Monitor” on page 78.

Scheduling a project

To schedule a project, select Tools | Runtime Manager. Use the Runtime Manager to create, edit, delete, execute, and terminate tasks.

See “Managing jobs and scheduled tasks” on page 57.

Managing jobs

Use a job to set up powerful control flows for one or more projects. You can schedule to run a Sybase ETL job without any user interaction.

Depending on the success or failure of a project within a job, you can control the job execution.

Job components

A job that executes a single project consists of at least:

- A Start component
- A Project component
- A Finish component

You can extend a job to include multiple:

- Projects in sequential or parallel order
- Synchronizers
- Finish and error components

A Start component is always followed by one or more Project components.

❖ **Creating a job**

1 In the Navigator, right-click Jobs and select New | Job. Available job components appear in the Component Store.

2 Drag the Start component from the Component Store to the Design window.

3 Add the Project component and connect it to the Start component.

4 Add the Finish component and connect it to the Project component.

5 Double-click the Project component.

6 Select the project you want to include in this job. Click Save.

The job is now ready to be executed in Sybase ETL Development or as a scheduled task.

From the Navigator, you can display and access the projects included in a job.

❖ **Modifying a job**

1 In the Navigator, double-click the job name, or right-click the job, and select Open.

2 Modify the job and save the changes.

❖ **Copying a job**

1 Double-click the job to copy to open it in the Design window.

2 In the Navigator, right-click the job and select Save As. Alternatively, select File | Save As.

If you are working with multiple repositories, select the target repository.

3 Provide a name for the job. A copy of an existing job is created, leaving the original untouched, and storing no references to the originating job.

Note Copying a job to a different repository does not copy the projects included in the job. You must select projects from the new repository for all project and multiproject components in your job.

❖ **Transferring a job**

If you are using multiple repositories, you can copy the complete job and projects included, from one repository to another, maintaining references to the originating objects. For example, you may want to move a job from a development repository to a test or production repository. By storing the references to its origins, the transfer recognizes the job the next time it is initiated and selectively replaces everything related to the incoming object.

1 In the Navigator, right-click the job to transfer and select Transfer. Alternatively, select File | Transfer.

2 Select the repository to where you are transferring the job.

The job and all included projects are copied from one repository to another, but the originating objects are referenced.

Note Transfer copies only job definitions. Related data, such as, parameter sets or execution properties, are not transferred.

❖ **Deleting a job**

1 In the Navigator, right-click the job and select Delete.

2 Click Delete. By default, only the selected job is deleted. To delete the job and all included projects, select the Delete Included Projects option.

Note Before you delete the projects used in a job, make sure the projects are not used in other jobs. This is not checked automatically. Projects that are currently open for design and locked by any user are not affected.

❖ **Renaming a job**

1 In the Navigator, right-click the job.

2 Select Rename.

Controlling job execution

You can control job execution by:

- Using the synchronizer component, which allows you to branch job execution based on a project's success or failure.
- Ignoring errors on each project.

See “Job components” on page 210.

Executing a job

You can execute a job directly from Sybase ETL Development, or at specific time intervals as a scheduled task of the operating system task manager.

- To execute a job currently open in the Design window, select Run | Execute.
- To execute a job directly from the Navigator, right-click the job, and select Job Execute.
- To schedule a job, select Tools | Runtime Manager. See “Managing jobs and scheduled tasks” on page 57.

Scheduling a job

To schedule a job, select Tools | Runtime Manager. Use the Runtime Manager to create, edit, delete, execute, and terminate tasks.

See “Managing jobs and scheduled tasks” on page 57.

Using templates to create projects and jobs

Use templates to automatically create projects and jobs.

Building a migration template using the template assistant

The template assistant lets you create a new template or use an existing template to migrate data from one database to another.

❖ **Building a migration template**

1 Select File | New | Template. Alternatively, right-click Templates in the Navigator and select New | Template.

2 Enter the migration details:

- Provide a name for the template. The name is used for the template object and, further qualified, for the generated transformation objects.
- The migration type should be DB to IQ.
- To use multiple engines for execution, select “Allow execution on multiple engines.”
- If you want to use multiple writers for loading data to the IQ database, select “Use IQ Multiplex”. Select this option if more than one table is being migrated to the IQ database.

  Note To support multiplex execution, you must install the Sybase SQL Server 11 ODBC driver on the same machine as ETL Development and ETL Server.

- To enable loading bulk load data into the IQ database from files located on remote host machines, select “Use IQ Client Side Load.”
- To lock the target table in Exclusive mode and prevent it from being updated by concurrent transactions, select “Use IQ Lock Table”. If selected, no other transaction can execute queries or perform any updates against the locked table. Use IQ Lock Table also queues multiple projects that load the same table in Sybase IQ.

  If you select this option, you must also specify the maximum blocking time that the project should wait before acquiring the lock.
- To enable transactionality for the generated job or projects, select Transactional. Data is committed at the end of the write operation for a successful execution, and rolled back for an unsuccessful execution.

  Note If the Transactional option is selected, all the data source and data sink components that support transactionality, are created with their Transactional property enabled.

- Click Next.

3 Enter connection details for the source database and select the tables to transfer. See “Database connection settings” on page 94.

Note The database connection properties are the same as for the DB components.

Click Logon to view the list of available tables for the specified database. By default, each table is selected for transfer. Unselect the tables you do not want to transfer. You can also choose one or more table rows, right-click and select Exclude. To include a table for transfer, right-click and select Transfer.

Alternatively, you can click one of these icons:

- Exclude all objects from transfer to exclude all tables.
- Include all objects in transfer to include all tables.

To view additional information about a table, choose the table row, right-click and select:

- Browse – to view table data.
- Count – to view the record count of the selected table. To view the record count for all tables, click Count All.

Click Next.

4 Enter database connection properties for the destination database. See “Database connection settings” on page 94.

Click Logon to view the list of available tables. To view the table data or the record count of the selected table, right-click and select Browse or Count. To view the record count of all tables, click Count All.

Click Next.

5 Specify transfer settings for tables to be transferred.

a Select Preserve schema/owner to retain the schema or owner information of the source table.

Note The same schema or owner must exist in the destination database.

b Enter stage properties.

In the Stage and Stage Server fields, specify the path for the load stage properties of the DB Bulk Load IQ component. If “Use Pipes” is selected, paths are automatically set. If Use Pipes is not selected, manually provide the values ended by the path delimiter. For example, *C:\ETLStage*.

See “DB Bulk Load Sybase IQ properties list” on page 170.

Note The Use Pipes option and the Stage server field are not available if you selected “Use IQ Client Side Load” in the migration details window in step 2.

c Select source attributes.

By default, all attributes of a table are selected for transfer. To change the attribute selection, click the icon in the Columns field.

In the Select Attribute window, unselect the attributes to exclude from transfer. You can also select one or more attribute rows, right-click, and choose Exclude.

d Select destination tables.

It is assumed that source and destination table names are the same. To use different names, enter a new name into the Destination field or select an existing table.

e Select additional options to perform appropriate actions for each table:

- Data model options – before the transfer starts, verify that the destination tables exist. The data model options can help you set up the destination data model. They do not affect execution, but affect the data model when it is created from the template.

  To create a new destination table based on the selected source attributes, select Create Table, or right-click the option and select Activate. To re-create an existing table, select Drop Table.

- Execution options – these options affect the execution on project level.

  Select Truncate to remove all records from the destination table before loading. This option corresponds to the Truncate Table property of the target component.

  The failure of a critical project causes the job to stop execution and signal failure. The Critical option and the Ignore Errors option correspond to the properties of the multiproject job component.

  The Ignore Errors setting does not affect the projects generated through this template.

6 Select the tasks you want to perform on the collected data.

Note Except for saving the template, you can alternatively perform all tasks described here, by right-clicking a stored template in the Navigator.

- Save template – store the template in the repository. Storing allows you to reuse the collected data for similar jobs.
- Build projects and jobs – create one project for each source table, and a migration job that controls the execution of all the projects.
- Create the destination data model – set up the destination data model according to the data model options you entered. Click Advanced to enter SQL commands, which are executed before the destination tables are created.
- Execute job – available only if Build projects and jobs is selected. If you select this option, the generated job is executed after migration template data is processed.

Note Select at least the Save template or Build Projects and jobs options to not lose collected data.

Click Finish.

Note Before you can execute the generated job, either register engines or open the job and deactivate the MultiEngine Execution option. See “Using multiple engines to reduce job execution time” on page 76.

While processing the data, you can view the current state and progress.

Managing a migration template

❖ **Creating a template**

1 In the Navigator, right-click Templates.

2 Select New | Template.

The template assistant guides you configuring a migration template.

❖ **Modifying a template**

1 In the Navigator, either double-click the template, or right-click it and select Open.

2 Modify and save the template.

❖ **Copying a template**

1 In the Navigator, right-click the template.

2 Select Copy. Enter a name for the new template. You can also copy a template into a different repository.

❖ **Deleting a template**

1 In the Navigator, right-click the template.

2 Select Delete.

Note Deleting a template does not affect jobs and projects that are based on that template.

❖ **Renaming a template**

1 In the Navigator, right-click the template and select Rename.

2 Enter a new name for the template.

❖ **Building a job from a template**

To create a migration job and all related projects based on a stored template:

- In the Navigator, right-click the template and select Build. To enforce unique names, a creation timestamp is added to all object names.

❖ **Creating a data model from a template**

To set up the destination data model according to the data model options stored with the template:

- In the Navigator, right-click the template and select Create Data Model.

Creating and simulating a sample project

A project usually contains one or more:

- Data providers that provide the data feeding the project data stream
- Data transformers that transform or remap field values
- Data sinks that write transformed values to their target

Note You can view results of this section in the Demo Getting Started project in your default repository.

For detailed information about components, properties and features, see Chapter 5, “Components.”

Adding a data provider

Use one of these methods to add DB Data Provider Full Load to your project:

- Drag the component from the Source tab of the Component Store to the Design window.
- In the Component Store, right-click the component that you want to add, and select Add.
- In the Component Store, double-click the component you want to add.

When you add a component to the Design window, the default configuration of the component is displayed.

Note Properties shown in **bold** in any configuration window are required.

❖ **Configuring the data provider**

1 Select ODBC from the Interface drop-down list. See “Database connection settings” on page 94 for information about the different interface types.

2 Select ETLDEMO_US from the Host Name drop-down list.

When you confirm the initial component settings, the settings appear in the Properties window.

3 To define the information to retrieve from the data source, click the query icon in the Query field.

4 Click the Query Designer icon to generate the query.

Note You can also enter the SQL query manually.

The left pane of the Query Designer window lets you navigate the table catalog of the connected database.

5 To add one or more tables, drag the table name onto the Design window, or right-click the table name and select Add Object to Query.

6 Click and drag the PRODUCTS table to the Design window.

7 Click Save to close Query Designer and return to the Query window. The select query is automatically generated.

8 Click Execute the Query icon to run or test the query.

9 Click Save to close the Query window.

Note When you have successfully configured a component, the color of the ports associated with it change from red or yellow to green.

Adding a data sink

Use one of these methods to add DB Data Sink Insert to your project:

- Drag the component from the Destination tab of the Component Store to the Design window.
- In the Component Store, right-click the component that you want to add, and select Add.
- In the Component Store, double-click the component you want to add.

As soon as you add a component to the Design window, the component displays its default configuration.

Note Properties shown in **bold** in any configuration window are required.

❖ **Configuring a data sink**

1 Select ODBC from the Interface drop-down list. See “Database connection settings” on page 94 for information about the different interface types.

2 Select ETLDEMO_DWH from the Host Name drop-down list.

3 Enter PRODUCTS in the Destination Table field. Alternatively, click the Destination table icon and select PRODUCTS.

4 Click Finish to confirm your settings.

Your project now consists of two components. The link between the components is created automatically if you selected “Create automatic links when components are added” in the File | Preference window. If the line is not automatically created, click on the OUT-port and drag it onto the IN-port of the data sink to create it.

The outgoing port (OUT-port) of the DB Data Provider Full Load component and the ingoing port (IN-port) of the DB Data Sink Insert component both display in green. This indicates that both components are configured.

In the Property window for the DB Data Sink Insert component, you can review and set all properties of the selected component.

❖ **Reviewing and defining attribute mappings**

1 Right-click the link between the components. The color of the link changes to green.

2 Select Mapping.

The mapping between the data source and the target source is created automatically. To change mappings, select the connecting line and attach it to another connection point.

Note You can map only to an unassigned target connection point. If all target connection points are already assigned, unassign a connection point by deleting the mapping line that is currently linking to it. Select the mapping line and press the Delete key, or right-click and select Delete.

Adding a Data Calculator

1 In the Component Store, click the Transform tab.

2 Select and drop the Data Calculator Javascript component onto the link connecting the existing components. The color of the link changes to blue.

After releasing the Data Calculator component:

- It is linked with the components to the right and left.
- The Data Calculator window appears.

The Data Calculator window has a Tabular and Graph view:

- Use the Tabular view to enter transformation rules.
- Use the Graph view to visually define the mapping sequence between the IN-ports and OUT-ports.

3 Click the Graph tab. The IN and OUT boxes represent the current structure of the port attributes.

4 Click Yes to assign a default mapping by order.

5 Click the Tabular tab to return to the tabular view.

6 Change all incoming data for the PR_NAME attribute into uppercase letters:

```
uUpper(IN.PR_NAME) ' OUT.PR_NAME
```

7 Enter uUpper(IN.PR_NAME) in the Transformation Rule column of the IN.PR_NAME attribute. The IN.PR_NAME value is forwarded to the OUT.PR_NAME attribute.

8 Click Save to confirm your settings. The green color of all ports in the project indicate that all components have been successfully configured.

9 Select File | Save.

Starting the simulation

1 To initialize all components, click the Start icon on the toolbar.

2 Click Step to step through the project from component to component.

At any point during the simulation, you can preview the current set of data, by right-clicking the link and selecting Preview. For example, when the first step executes, the data records are forwarded from the source component to the data calculator. A number on the link indicates the number of records transferred.

Note The Preview option is not available when there are no processed records or when no simulation data is available.

3 When all data has been processed, select one of these:

- Execute post-processing as for successful execution – to commit all tasks performed by the transactional components and to reset the project to its initial state.
- Execute post-processing as for failed execution – to roll back all tasks performed by the transactional components and to reset the project to its initial state

Click Yes to confirm resetting of the interactive trace. This clears all port buffers, releases temporary tables, and closes all database connections and temporary files.

If you click No, all open database connections and port buffers are retained. You can inspect, reconfigure, and then step through individual components again.

CHAPTER 4

Advanced Concepts and Tools

Query Designer

Use the Query Designer to:

- Browse the table catalog of any connected database of the current project
- Create SQL queries by using a graphical user interface
- Review generated SQL statements
- Execute SQL queries against the database
- Browse data in a selected table or view
- Create a table in the schema

- Delete all records in a table

Note To delete all records in a table, select "Enable delete functionality of database objects" in the File | Preference window. See "Customizing preferences" on page 19.

- Count the number of records in a table or view

Opening the Query Designer

This section uses the Demo Getting Started project from the Demo Repository to create queries using the Query Designer.

To open the Query Designer:

1 In the Navigator, double-click the Demo Getting Started project to open it in the Design window.

2 Double-click the DB Data Provider Full Load component. Alternatively, select the component to display its properties in the Properties window.

3 Click the Query icon.

4 Click the Query Designer icon.

Query Designer interface

The Query Designer interface consists of:

- Query Definition pane – includes the Design window, which is used to automatically generate select statements that are specific to the records you are working with.
- Navigator – displays all the tables and views on the Model tab, and the recently used tables or views on the Recent tab. You can set the default number of tables or views that you want to view under the Recent tab, in the File | Preferences window. See "Customizing preferences" on page 19.
- Attribute tabs (Select, Join, Where, Sort, Group) and a Generated Query tab – allow you to view the attribute details, as well as the generated query at any point of the query creation.

Creating queries

This section uses the Demo Getting Started project from the Demo Repository to create queries.

❖ Creating a simple query

To generate a simple query that retrieves all attributes from a table, use the PRODUCTS table.

1 In the Navigator, select the Model tab, then click the table or view name. To search for a particular table or view name, press Ctrl+F.

2 Drag the selected object to the Design window.

3 To view the results of the generated query, select View | Generated Query or select the Generated Query tab.

❖ Creating a query using multiple tables

To generate a query that joins and retrieves information from two tables, use the PRODUCTS and SALES tables.

1 Drag the PRODUCTS table from the Navigator to the Design window.

2 Drag the SALES table from the Navigator to the Design window.

3 Create a join between the tables by linking the PR_ID fields of both tables. If you want Query Designer to automatically create joins between identically named attributes within tables or views:

   a Select File | Preferences from the main Sybase ETL Development window.

   b Select Workbench | Query Designer, then select Create joins automatically. See “Customizing preferences” on page 19.

4 To view join attribute details, click the Join tab in the Query Designer window.

❖ Modifying the default settings of a join

A join between two tables is indicated by a line that connects the joining fields. The line is labeled with a join operator, which, by default, is called Equi Join.

1 Right-click the line connecting the two joining fields.

2 Select Modify.

3 Select a join type.

❖ **Modifying the sort order of joins**

1 In the Join tab, right-click a row and select:

- Move to start
- Move up
- Move down
- Move to end

2 To revert to the default state of the join at any point, right-click a row and select Sort joins to default order.

❖ **Adding one or more attributes to the select clause**

1 Drag the PRODUCTS and SALES tables to the Design window, if they are not there already.

2 To add a single attribute, right-click the attribute you want to add, and select Add Items to Selection. To add more than one attribute, hold down the Ctrl key while you select the attributes you want to add.

Alternatively, click the PRODUCTS and SALES tabs in the Query Definition pane and select the attributes you want to add to the select clause. To search for an attribute, click the Search icon and provide your search criteria.

You can use an asterisk (*) as a wildcard to search for any number of unknown characters. For example:

- If your attribute is an integer datatype, your search criteria can be one of these:

```
int, int*, i*ger
```

- If your attribute name contains “PROD” and has “CD” at the end, your search criteria can be:

```
*PROD*CD
```

❖ **Adding all attributes of a selected table to the select clause**

1 In the Query tab, click the header of the table.

2 Right-click and select Add Items to Select.

❖ **Viewing attribute details and generated query**

1 To view a query generated by the Query Designer, select View | Generated Query, or select the Generated Query tab.

2 Click the appropriate tabs to view the attribute details.

❖ **Adding functions to the select attribute**

1 In the Select tab, right-click the attribute to which you want to add functions.

2 Choose the functions to add.

Note To enforce appropriate and reliable attribute names for viewing data and port structures, define alias names for all attributes to which functions have been applied. See Adding alias names to selected attributes and "Use column aliases when entering queries" on page 324.

❖ **Adding alias names to selected attributes**

- In the Select tab, enter a name in the Alias column to enforce an output column name used in Data Viewer and associated port structure.

Content Explorer

Use the Content Explorer to browse schema information and data content of all connected data sources. Use the Content Explorer to generate ad hoc queries, which cannot be saved to a file or to the repository. To save generated SQL statements, select and copy the generated query from the Generated Query window.

Use one of these methods to open the Content Explorer:

- Select Tools | Content Explorer. The Choose Data Source window displays all the components currently connected to data sources. The names in the list of currently connected databases is a combination of a user-defined name and the generic name of the component type. Select a component and click Start to open the Content Explorer.
- Right-click a database component and select Content Explorer.

File Log Inspector

Use the File Log Inspector to view information about project and jobs execution, fatal errors, and to review the system log. Log files are located in the *log* subdirectory of the installation directory.

1 Select Tools | File Log Inspector.

2 Click the log file information you want to view. You may see one or all of these log files:

- *alert.log* – captures history of the triggered alerts. Provide alert details such as, alert name, event type, date and time, and alert message.
- *execution.log* – captures information about job and project execution.
- *system.log* – captures information about system activities, and operational and exceptional events. You can check the error codes in the *system.log* file to determine the reason and possible solution for errors encountered while working with Sybase ETL. The error codes that you may see in the *system.log* file include:

Error code	Type	Description
0	Information	Job or project execution is successful
100 or 110	Error	ETL engine initialization error
101	Error	Invalid license error
1100	Error	ETL exception failure, including incorrect command line usage
1103 or 1104	Error	Failure due to an unspecified exception
10001	Error	Failure to retrieve information from repositories, including jobs, projects, and parameter sets
10005	Error	Job execution failed
10006	Error	Project execution failed
10007	Error	Project execution ending in a rollback state
10101	Error	Connection to the repository database failed

The detail level of data written to *system.log* depends on the trace level that has been set.

To set the trace level:

- Select Tools | Enable System Trace.

- Use the uTracelevel(*n*) function in a JavaScript procedure.

  uTracelevel(*n*), where *n* is a value of 0 through 5, lets you set the trace level from within a component.

- Specify the trace level in the *default.ini* file in the *etc* subdirectory of the installation folder by modifying this line:

  ```
  Tracelevel=value
  ```

  where *value* is the trace level you want to set. You must restart Sybase ETL for the change to take effect.

Note Sybase recommends you to set a trace level to either 0 or 5.

3 (Optional) Click "Truncate log" icon on the toolbar to truncate the log. Click Yes to confirm.

4 (Optional) Click the "Select rows containing a string or regular expression" icon on the toolbar to locate a particular log file.

Note The Log File Inspector displays only the last 1MB of a log file. To view older records of a large log file, use a text editor to open the file.

Managing jobs and scheduled tasks

Use the Runtime Manager to manage projects and jobs, and to see an overview of your current scheduled tasks. Use the Runtime Manager scheduling wizard to manage scheduled tasks.

The Runtime Manager is based on the ETL Scheduler, which enables you to create, edit, delete, execute, list, import, and terminate scheduled tasks using a selected engine. Select the engine you want to use to manage your scheduled tasks from the Schedule Service drop-down list. A list of all tasks scheduled on that engine is displayed.

Note You can schedule tasks only on grid engines running in the same subnet as ETL Development. To schedule tasks on grid engines running in multiple subnets, run an instance of ETL Development in the subnets in which these engines are running.

❖ **Creating a new schedule**

1. Select Tools | Runtime Manager.
2. Select Actions | Create. Alternatively, click the Create a New Schedule icon on the toolbar.
3. Select the project or job to execute. If required, select a parameter set or specify the Rep CDC Instance Name needed for incremental loading. Click Next.
4. Enter the schedule details:
   - Enter the schedule name, which must be unique.
   - Provide the start date and time for the schedule.
   - Click Repeat Task to specify how often you want the task to be repeated.
   - Click Advanced to specify:
     - Execute new task concurrently (default) – allows running multiple instances of the task concurrently.
     - Execute new task sequentially – waits for completion of the current process before executing the new one.
     - Do not execute new task – continues processing the current task and ignores the request to start a new task.
     - Cancel the running task before executing new task – stops the current process and starts the new one.
     - The “Stop the task after” option terminates a task that has been running for more than the specified duration. By default, this option is disabled.
   - Provide the end date, after which the schedule is inactivated.
   - Specify how often you want the task to be executed:
     - Daily – at a specified day interval.
     - Weekly – at a specified week interval.
     - Monthly – at a specified time on a particular day of each selected month. Specify the days of the month, and select the appropriate calendar months.
     - Once – only once at a specified date and time.
     - At engine startup – each time the engine starts.

Click Next.

5 If the schedule is successfully created, a message appears. Click Finish.

The new schedule appears in the Runtime Manager along with the existing schedules, if any.

❖ **Editing a schedule**

1 Select the scheduled project or job to edit.

2 Click the Edit a Schedule icon on the toolbar, or select Actions | Edit. Alternatively, you can double-click the scheduled project or job to modify.

❖ **Executing a schedule**

1 Select the scheduled project or job to execute.

2 Click the Execute a Schedule icon on the toolbar, or select Actions | Execute.

Note A scheduled task uses the same performance logging level that has been set in the Preference window. To use a different log level, change the "Performance Logging" option before creating or editing the task, and then reset it after saving the task. See"Collecting performance data" on page 80.

❖ **Deleting a schedule**

1 Select the scheduled project or job to delete.

2 Click the Delete a Schedule icon on the toolbar, or select Actions | Delete.

❖ **Setting refresh options for schedules**

1 Select Actions | Enable Auto-Refresh to periodically update the schedule information. By default this option is not selected.

Note If Auto-Refresh is not enabled, the information displayed may be old. Refresh the schedule manually to view the current status of tasks.

Select Action | Refresh Interval to define the intervals at which you want the schedules to be updated. The refresh interval value should not be less than 2 seconds.

2 Click OK.

❖ **Importing tasks from the Windows scheduler to ETL scheduler**

You can import tasks previously scheduled on the Windows Scheduler to the ETL Scheduler.

Note You can only import scheduled tasks created in Sybase ETL 4.8 or earlier.

1 From the Schedule Service drop-down list on the toolbar, select the target engine to which you want to import the Windows scheduled tasks.

2 Select Actions | Import.

3 Select the source engine from which to import the scheduled tasks and click OK. Windows scheduled tasks on the selected engine are imported to the ETL Scheduler.

4 Tasks are imported using the original name unless a task with the same name exists on the ETL Scheduler. If the current name already exists, you must provide a new name.

Note The name of the scheduled task must be unique.

5 Click Yes to delete the Windows schedule.

Click No, if you want to import the tasks from the source engine to any other engine before deleting them, or if you want to manually delete the tasks later, using the Windows Task Scheduler.

6 You see a message showing the result of the import. Click OK.

❖ **Terminating a schedule**

1 Select the scheduled project or job to terminate.

2 Click the Terminate a Running Schedule icon on the toolbar, or select Actions | Terminate.

Customizing SQL and transformation rules

When setting up a project or job, you can customize:

- SQL queries for setting up the source components

- SQL commands for processing tasks
- Expressions, conditions, and procedures to manipulate the transformation process

The format of SQL commands depends on the database system that is connected to the component. However, the format of the transformation language supported by Sybase ETL (JavaScript) does not change, regardless of the source or target system you use in your projects.

You can include JavaScript expressions in Square Bracket Notation (SBN), which considerably reduces your customization efforts. SBN expressions can be part of component properties (if the Evaluate option is activated for the specific property), like SQL statements, or any processing commands. An SBN expression is evaluated at runtime, as opposed to constant values, which are defined at design time. See “Evaluating SBN expressions” on page 88.

Expressions and procedures

An expression is a combination of identifiers and operators that can calculate a single value. A simple expression can be a variable, a constant, or a scalar function. Operators can be used to join two or more simple expressions into a complex expression.

Examples of expressions are:

```
'Miller'
uConcat("Time ", "goes by")
(uMid(SA_ORDERDATE, 1, 10) >= '1998-01-01')
```

A procedure is a programming unit that includes expressions, statements, and control structures. You can write procedures in JavaScript, for example:

```
if (IN.PR_PRICE < 250)
       OUT.PR_GROUP2 = 'low end' ;
else {
    if (IN.PR_PRICE < 1000)
       OUT.PR_GROUP2 =  'mid range';
    else
       OUT.PR_GROUP2 = 'high end';
}
```

Variables

A variable is a symbolic name for a value. There are two basic properties of a variable:

- Scope, which specifies the range of the environment in which the variables can be referenced.
- Datatype

There are two kinds of variables:

- Port variables
- Component variables

Port variables

The values of the port structure are referenced as port variables within a component. There are automatic port variables for both IN-ports and OUT-ports. Port variables are valid within the component, and they inherit the name and datatype of the port structure. The name of the variable is prefixed with IN. for the IN-ports and OUT. for the OUT-ports. IN-port variables are read-only, but OUT-port variables can be written.

This example uses port variables in an expression:

```
uUpper(IN.CU_NAME)
```

This example uses port variables in a procedure:

```
OUT.CU_NAME = uUpper(IN.CU_NAME);
```

Component variables

Component variables are associated with component properties and represent the current evaluated property content. You can reference these variables inside the component. The name of the variable is prefixed with “REF.” For example:

```
uIsNull(REF.Host)
```

To provide flexibility in transformations, all port and component variables internally use the datatype string. If you use numeric values, this may result in unexpected behavior.

If multiplied by 1, the numeric value of a string variable is used in a calculation:

```
IN.Margin="2", IN.Price="10"
IN.Margin>IN.Price  - returns TRUE
```

To enforce a numeric comparison, use:

```
IN.Margin*1>IN.Price*1  - returns
FALSE
```

Do not use any of these reserved JavaScript keywords for port and component variable names:

Reserved JavaScript keywords

break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile	const	export		

Functions

Sybase ETL includes a complete set of functions and operators that supports Unicode character sets.

You can recognize Sybase ETL functions by the prefix u, for example, uConcat().

Square Bracket Notation

Most component properties can contain SBN expressions that are evaluated before being used by the Sybase ETL Server on project configuration, simulation, or execution. An SBN expression is a JavaScript expression surrounded by square brackets [..]. Multiple SBN expressions can be used within a single string value.

You can use SBN expressions in component properties like:

- SQL queries
- Processing SQL statements
- Transformation rules
- File names

- Path definitions
- URLs

Examples

A literal is a string surrounded by quotes. If you use SBN in a literal, the SBN is evaluated first.

```
'Arrival Date: [uDate('now', 'localtime')]'
```

This expression specifies the path of a file in the Text Data Provider:

```
[uSystemFolder('APP DEMODATA')]\PRODUCTS.TXT
```

Note In the Property window of components, the Eval column indicates whether a value entered is evaluated to resolve SBN expressions. For many property items, the Eval column is optional. To toggle the Eval check box, right-click the property item and select Evaluate.

Working with SQL properties

Nearly all components with database connectivity support custom SQL statements executed in different phases of the transformation. There are two basic types of SQL properties, Queries and Scripts. For both types:

- Any SQL accepted by the connected database system is allowed. Using SQL92 allows you switching to different database systems without changing the statements.
- SBN expressions are allowed. See “Square Bracket Notation” on page 63.

Queries

SQL queries are used for all components that extract data, mainly the Data Provider and Staging components. The columns of the query result set define the respective OUT-port structure.

The SQL query result set must contain at least one column. Example of a SQL query:

```
SELECT cu_no, cu_name FROM customers
```

Scripts

A SQL script consists of one or more SQL statements not returning any data. For example, there are properties allowing you to execute SQL statements during initialization (pre-processing) of a component or after completion (post-processing) of the project.

Some considerations:

- SQL statements must not return output after execution.
- You can enter multiple SQL statements in a SQL property by using a semicolon as a statement delimiter.
- When using stored procedures in a SQL property, include the call keyword before the stored procedure name. For example, call my_proc(); where my_proc() is the name of the stored procedure.

Example of a SQL script:

```
DELETE FROM products;
UPDATE customers
SET cu_desc = 'valid';
```

Using the SQL Property window

To modify the value for a SQL property of a component, click the respective Edit icon in the Properties window. In the SQL Property window, you can:

- Manually enter statements
- Use the Query Designer. See “Query Designer” on page 51.
- Look up the database schema.
- Run a query or script.
- Save the SQL property value.

Entering SQL statements

You can enter the SQL statements in the text field.

Creating a query

You can open the Query Designer to graphically construct a query. See “Query Designer” on page 51.

In the Query Designer, select View | Generated Query or click the Generated Query tab, copy the entire or parts of the generated query, and paste it in the Query text field of the SQL Property window.

Note For Query properties, the Query Designer provides a Save button. If you click it, the entire content of the Query text field is replaced with the generated query.

Looking up the database schema

You can select tables and attributes for your statements from the associated database schema by clicking the Database Lookup icon. In the Database Lookup window, select the tables and attributes you want to use in the statement and click OK.

Executing SQL statements and viewing query result sets

To execute the current SQL statements click the “Execute” icon. For Query properties, you are prompted to specify the number of records to view. The result is displayed in the Data Viewer.

Using SBN expressions

This section shows how to use SBN expressions in SQL properties. Although the following example is a Query, the technique applies to both types.

Assume you want a SELECT statement to retrieve a specific customer record.

- You might include a constant customer number for that record:

```
select * FROM CUSTOMERS WHERE CU_NO = '12345678'
```

- With SBN, you can use a more flexible approach by assigning the constant value of CU_NO to a custom component property. See “Custom properties” on page 89. Assuming that value ‘12345678’ was assigned to a property CustNo, the select statement with the dynamic expression looks like:

```
select * FROM CUSTOMERS WHERE CU_NO = '[REF.CustNo]'
```

- You can use any of the Sybase ETL functions inside the SBN expression. The following statement returns the same record using a value of “1234” for CustNo1 and a value “5678” for CustNo2:

```
select * FROM CUSTOMERS WHERE CU_NO = '[uConcat
(REF.CustNo1, REF.CustNo2)]'
```

Using the JavaScript Editor and Debugger

JavaScript is an object-oriented scripting language designed that is embedded into other products and applications. JavaScript allows manipulation of objects to provide programmatic control over them.

JavaScript functionality is enriched by grid functions, which enhance the flexibility of the language. The JavaScript Editor and Debugger let you interactively edit, debug, and execute JavaScript code.

The JavaScript Editor and Debugger is mainly used (but not restricted) to set up transformation rules on incoming data. Inside the JavaScript Editor and Debugger, you can execute and test the scripts using a single input record.

The JavaScript Editor and Debugger offers:

- Color-coded syntax for better readability
- A watch list to control the assigned values of variables and attributes while you are running or stepping through code
- Multiple user-defined breakpoints to stop code execution at any line positions
- User-defined Go points to arbitrarily choose the position from which code executes
- Step mode to execute code line by line
- Step-over during debugging
- Evaluation of JavaScript expressions
- Verification of the result of code execution

Starting the JavaScript Editor and Debugger

1 Double-click the Data Calculator JavaScript component, or click the Rule icon in the Property window.

2 Select a row in the Transformation Rule column and click the Edit icon.

The JavaScript Editor and Debugger window includes:

- Navigator – consists of the Variables tab and the JavaScript tab. The Variable tab consists of IN-port and OUT-port variables, temporary, and predefined variables. The JavaScript tab includes all functions, commands, and system variables that can be applied within the procedure.
- Edit/Debug pane – lets you edit the actual code.
- These tabs appear at the bottom of the window:
  - Tasks – displays the results of the validation after your procedure has been compiled.
  - Watch List – displays selected variables and their values when stepping through the code during debugging.
  - Input Records – displays the content of the current input record. To synchronize input records and output records, click the Start debugging icon on the toolbar.
  - Output Record – displays the content of the current output record.
  - Expression – displays the result of the expression after you enter a JavaScript expression and click Evaluate on the toolbar.

Note The Evaluate button is disabled when you are not in an active debugging session.

Edit and Debug mode

When launched, the JavaScript Editor and Debugger opens in edit mode. To switch to the debug mode, select Debug | Start.

A dark grey background of the edit area indicates the debug mode. To switch to edit mode, click the “Stop debugging and start editing” icon on the toolbar.

Editing and debugging JavaScript

To validate JavaScript code, select Debug | Start. The result of the validation appears in the Tasks tab

The JavaScript Editor and Debugger offers efficient features to trace the execution of a script. You can step through a code line-by-line or step through from one breakpoint to another. You can check the current value of a variable at any time.

Note A comment line starts with two forward slashes // at the beginning of the line.

❖ **Stepping through the code**

The JavaScript Editor and Debugger works without input data at the IN-port of the component. However, for best results, populate the IN-port with data before using the debugging features.

1 Validate the script or switch to the debug mode.

A green arrow, pointing initially to line 1, indicates the progress of the execution while stepping.

2 Make sure the result message in the Task tab displays “Successful compilation.”

3 To move to the next line, click the Step icon on the toolbar.

At any point during stepping you can inspect the variable name and the current value. To do so, select the variable in the Navigator and right-click.

❖ **Add and removing breakpoints**

Instead of stepping through the procedure line-by-line, you can include breakpoints at selected lines.

1 Click the line where you want to set or remove a breakpoint.

2 Right-click and select Add/Remove breakpoint.

❖ **Stepping to a breakpoint**

1 Click the Go icon on the toolbar for each step.

2 Click the Go icon on the last breakpoint to execute the rest of the script.

Monitoring values in the watch list

You can use the watch list to monitor the changes of variable values during the code execution. When stepping through the code, you can see any change that occurs to one or more variables in the Watch List.

❖ **Adding a variable to the watch list**

1 Right-click the variable.

2 Select Add to Watchlist.

❖ **Removing a variable form the watch list**

1 In the Watch List tab, select the variable row and right-click.

2 Select Remove Watch Variable.

Special JavaScript features

To interrupt JavaScript execution, click the “Cancel a running script” icon on the toolbar from inside the editor.

Creating user-defined errors

To enforce an error and interrupt the execution of the project, use the throw("*xx*") function. For example, to stop execution if the name of a product (PR_NAME) exceeds the length of 20 characters, use:

```
if (uLength(IN.PR_NAME) > 20) (
      throw(“Product name exceeds maximum length”);
)
```

Creating user-defined functions

You can define functions inside a script and create nested functions. For example, this script results in a value 6 for variable *b*:

```
var a = 2;
var b = 20;
b = IncA(a);
// end
function IncA (a)
{
     var b = 3;
     a = IncB(b) + a++;
     return a;
     function IncB(b)
     {
          b = b + 1;
          return b;
     }
}
```

Converting datatypes

All variables in Sybase ETL are represented as strings. If you work with numeric values, this may result in unexpected behavior. You can use the functions parseInt() and parseFloat() to convert a string to an integer or a float, for example:

```
var a = "123";
var b = "22";

a > b

will return FALSE while

parseInt(a) > parseInt(b)
returns TRUE.
```

Including files

To include external files into a script, use the uScriptLoad("*filename*") function. The external file can contain any valid JavaScript constructs, including functions, thus allowing a kind of reusable code, for example:

```
uScriptLoad("C:\scripts\myfunc.js");
var a = 11;
var b = 2;
var c = 0;
b = gcd(a, b);
// gcd function defined in C:\scripts\myfunc.js
```

Executing SQL queries and commands

You can enter and execute a SQL query or a series of SQL commands for databases associated with Source, Lookup, Staging, and Destination components.

❖ **Executing SQL queries**

1 In the Design window, right-click the component and select Execute SQL Query.

2 Enter a select statement in the Query field. You can use any valid SQL notation of the connected database to build the query. To create the query using tables and views from the available database schema, click the Database Lookup icon.

3 Click the Execute the query icon.

On successful execution, the result appears in the Data Viewer window.

❖ **Executing SQL commands**

1 In the Design window, right-click the component and select Execute SQL Commands.

2 Enter the SQL commands in the Command field. To create the command using tables and views from the available database schema, click the Database Lookup icon.

3 Click the Execute the command icon.

A message appears on successful execution of the SQL commands.

Note Executing a series of commands does not return a result set.

Parameter sets

When you execute a project, all component properties are initialized with the values stored in the repository. Parameter sets provide a way to overwrite some of these values. For example, you can use parameter sets to change database connection settings when you move projects or jobs from development to production. To use parameter sets, you:

- Select the component properties you want to use as parameters.
- Store sets of parameter values.
- Assign a stored parameter set on execution.

❖ **Selecting component properties as execution parameters**

1 In the Properties window, right-click all component properties you want to use as execution parameters, and select External.

2 Right-click all component properties you want to assign a dynamic value using SBN expressions via a parameter set, and select Evaluate. Include all nonprinting values, such as, Tab, CRLF, and so on.

3 Save the project.

Note Provide unique names for at least all components that provide project parameters. You may also want to change the prompt and description of the properties. See “Custom properties” on page 89 for information on how to modify the prompt and description of properties.

Managing parameter sets

You can assign parameter sets to projects and jobs.

To open the Parameter Set window, right-click a project or job in the Design window or in the Navigator, and select Parameter Sets. The Parameter Set window displays a list of defined parameter sets for the selected project or job.

Note There are some properties where the values displayed on project design may differ from the values that you must provide in a parameter set. For a list of these properties and their values, see “Special property values” on page 74.

❖ **Creating a parameter set**

1 In the Parameter Set window, click Set | New. The window displays a list of the defined parameters and their current values at the bottom.

2 Overwrite the current values with the values you want to add.

3 Click Save.

4 Enter a name for the parameter set.

5 Click OK.

❖ **Modifying a parameter set**

1 In the Parameter Set window, choose a parameter set.

2 Click Set | Open.

3 Overwrite the current parameters with the new values.

4 Click Save.

❖ **Deleting a parameter set**

1 In the Parameter Set window, choose a parameter set.

2 Click Set | Delete.

❖ **Copying a parameter set**

1 In the Parameter Set window, choose a parameter set.

2 Click Set | Copy.

3 Name the new parameter set.

4 Click OK.

❖ **Executing a project or job with parameter sets**

1 In the Navigator, right-click a project or job.

2 Choose Execute Project with Parameter Set or Job Execute with Parameter Set.

3 Select a parameter set from the list or add the parameter values for a single execution.

4 Click Execute. See “Execution Monitor” on page 78.

Note If no stored parameter set is available, the window automatically opens in edit mode.

Assigning parameter values

To select a single parameter, click the appropriate list row. To select multiple parameters, drag the mouse over the respective rows, or use CTRL+click to select additional rows.

After you select a parameter:

- Enter the new value. The old value is overwritten.
- Edit the existing value directly in the Value cell and click Save.

❖ **Assigning the same value to multiple properties**

Because parameter sets are based on component properties, you may want to assign the same value to multiple properties.

1 Select the parameters whose values you want to assign.

2 Enter the new value and confirm that you want to use the value for all selected lines.

Alternatively, click the Edit Selected values button or right-click and select “Edit selected”. Edit, then confirm, the value.

Sorting the parameter list

You can use single or multiple columns to sort the parameter list.

❖ **Sorting parameters by a single column**

1 Click the column header.

2 Click multiple times to toggle between ascending, descending, and the original sort order.

❖ **Sorting parameters by multiple columns**

1 Click the primary column header multiple times to sort in the appropriate order.

2 Press Ctrl and click the secondary column header to sort the columns.

Special property values

There are some properties where values appear on project design differently from the values that you must provide in a parameter set.

Check boxes

For properties represented by a check box on project design, you must provide 0 (deactivated) or 1 (activated) as the value in a parameter set.

Expressions

To use dynamic values in a parameter set, enter SBN expressions in the same way as in the Design window. The Eval column indicates whether a property is enabled for expressions. See “Square Bracket Notation” on page 63.

You especially need expressions when setting values containing nonprinting characters, such as, Tab, CRLF, and so on. You must set the Evaluate option for these properties when designing the project.

Note You cannot validate expressions in the Parameter Set window.

Drop-down menus

Some menus do not display the underlying parameter value. Some of these values require you to set the Evaluate option to assign them via a parameter set.

Use this table to identify which value (Value) corresponds to the displayed one (Prompt) and whether you must enable Evaluate.

Component	Property	Prompt	Value	Evaluate
DB components	Interface	ODBC	dbodbc	
		Sybase	dbsybase	
		Oracle	dboracle	
		IBM DB/2	dbdb2	
		SQLite Persistent	dbpersistent	
		OLE DB	dbole	
Text components	Row Delimiter	Position		
		LF	[uChr(10)]	x
		CR	[uChr(13)]	x
		CRLF	[uConcat(uChr(13),uChr(10))]	x
	Column Delimiter	Position		
		Tab	[uChr(9)]	x
		Comma	,	
		Semicolon	;	

Component	Property	Prompt	Value	Evaluate
	Column Quote	None		
		Single Quote	'	
		Double Quote	"	

Using multiple engines to reduce job execution time

The grid architecture reduces job execution time by using parallel execution of projects on multiple distributed engines.

To leverage this scalability, you must:

- Install multiple grid engines
- Register your grid engines
- Prepare jobs for multi-engine execution

Note The terms “grid engine” and “ETL server” are used interchangeably in this guide.

Registering grid engines

After you install grid engines, you can register them for a special repository. When executing a multiengine job from that repository, all projects are executed on those engines.

To register grid engines, select Tools | Engine Manager. If connections to multiple repositories are open, select one of them. The Engine Manager window displays a list of engines that are already registered for the selected repository.

The properties of a registered engine are:

- Name – user-defined name for the engine.
- Host – name or IP address of the engine host.
- Port – number of the port the engine is listening on.
- Base Rank – user-defined ranking for the engines. A job first tries to executes the projects on the highest ranked engines.
- Description – description for the server.

You can manually register individual or multiple ETL servers.

❖ **Manually registering a grid engine**

1 Select Engine | New, or click the New Insert icon.

2 Enter values.

3 Click OK.

❖ **Registering multiple engines**

1 Select Engine | New Network Search. The New Engines window appears, displaying registration and additional information, as well as the online state of the engine.

2 Select the engine to register.

3 Click Add.

Note Click the Refresh icon on the toolbar or press the F5 key to update and reload the list of grid engines at any point.

❖ **Modifying an engine registration**

1 Select the engine in the list and choose Engine | Edit, or click the Edit selected engine icon. Alternatively, double-click the engine in the list.

2 Rewrite the current values with new values.

3 Click OK.

❖ **Deleting an engine registration**

1 Select the engine you want to delete.

2 Select Engine | Delete, or click the Delete selected engine icon on the toolbar.

Defining multiengine jobs

The parallel grid architecture lets you run a job on multiple engines. A typical multi-engine job contains multiple projects with few or no dependencies between them. Thus, projects can be executed on multiple engines at the same time.

1 Double-click a job.

2 In the Design window, right-click and select MultiEngine Execution.

During execution, the job uses the registered engines to distribute the projects.

Executing multi-engine jobs

To execute multi-engine jobs, right-click the job in the Navigator, and select Job Execute. Alternatively, use the Runtime Manager to schedule it.

Engine Monitor

The Engine Monitor displays information about all available grid engines in the environment, whether or not they are running or registered in the Engine Manager.

Note You can use only engines that are registered in the Engine Manager for multi-engine job execution.

To view information about the available grid engines, select Tools | Engine Monitor. The Engine Monitor displays engine details as well as information such as, the number of projects that are currently running and the number of projects that can be executed concurrently, on the grid engine. Use the Update Interval field to specify the number of seconds to wait between two updates. The default value is 5 seconds.

Execution Monitor

The Execution Monitor displays the properties of the current job or project:

1 Select the project you want to execute.

2 Click Execute on the toolbar.

The Execution Monitor window is divided into three panes, Job, Projects and Events.

The top part of the Execution Monitor window displays properties of the currently executing job.

Property	Description
Name	Job or project name
State	Current job execution state
Start	Start date and time
Stop	Stop date and time
Message	Error message

The Projects list contains a line for every project in the job.

Property	Description
Name	Project name
State	Current project execution state
Start	Start date and time
Stop	Stop date and time
Engine Name	Name of the executing engine
Engine Host	Host of the executing engine
Engine Port	Port of the executing engine
Message	Error message

The Event list displays the events and alert details.

Property	Description
Event	Event name
Alert	Alert name
Time	Start time, finish time, or error time
Message	Alert defined detailed message, e-mail preferences, subject, and body content.

Saving or copying the execution results

To save the results of the project you are executing to an HTML file, click Save Results.

To copy and paste the execution results of the job or project to a different application, such as Notepad or Microsoft Word, right-click the job or project in the Execution Monitor, and select Copy.

Cancelling job execution

To cancel job execution, click Cancel Execution on the Execution Monitor window.

Grid engines try to cancel running projects. Projects waiting to be executed are not started.

Analyzing performance data

While executing jobs and projects, Sybase ETL collects performance relevant data and stores it in a repository table.

❖ **Collecting performance data**

1 To collect and store performance relevant data to a repository, select File | Preferences | Performance Logging.

2 From the Logging Level list, select 1.

Viewing performance data

To view performance data overview of a selected project or job, in the Navigator, right-click a project or job and select Performance Data. Alternatively, open the project, right-click anywhere in the Design window, and select Performance Data.

The Performance Data window appears, displaying the execution details for the selected project or job under an Overview tab. By default, the performance data of executions performed within the last month are displayed. To change the range of execution dates that are displayed, change the values of the Execution Date Range on the toolbar.

Note You can view performance data of one project or job at a time.

To permanently remove performance data of any execution, select the specific row and click the “Delete selected performance log entries” icon on the toolbar. Alternatively, press Ctrl+D.

Warning! You cannot recover deleted performance data.

Viewing project performance data

To view project performance details, select a row in the Overview tab and click the “Drill down in performance data” icon on the toolbar. Alternatively, double-click the selected row or the corresponding bar in the chart displayed. A new tab displays details, such as component name, duration, records read, and records written.

Note When you view the performance data of a project using the IQ Loader DB via Insert Location component or the IQ Loader File via Load Table component, the performance table may not display the correct value for the number of records read and written. IQ loads data directly from files or remote databases; therefore, this information is not available to ETL. Information is available in the IQ Message log.

To view performance details for a selected component, select a row in the tab displaying the component details, and click the “Drill down in performance data” icon on the toolbar. Alternatively, double-click the selected row or the corresponding bar in the chart. The component performance details such as event name, duration, records read, and records written appear.

To return to the higher level of performance details, click the “Drill up in performance data” icon on the toolbar. To go back to the Overview tab, click the “Return to performance data overview” icon on the toolbar, or select Navigation | Return to Overview.

View job performance data

Select a row in the Overview tab and click the “Drill down in performance data” icon on the toolbar. Alternatively, double-click the selected row or the corresponding bar in the chart displayed. A new tab displays details, such as project name, duration, records read, records written, result, and load time.

To view performance details for a selected project, select a row, and click the “Drill down in performance data” icon on the toolbar. Alternatively, double-click the selected row or the corresponding bar in the chart displayed. The project performance details such as, component name, duration, records read, and records written are displayed.

To return to the higher level of performance details, click the “Drill up in performance data” icon on the toolbar. To go back to the Overview tab, click the “Return to performance data overview” icon on the toolbar, or select Navigation | Return to Overview.

Note To find the selected component in a project, select Tools | Show component. Alternatively, click the Show selected component icon on the toolbar.

❖ **Searching for performance data**

1 Select any row in the performance data table.

2 Press Ctrl+F to open the search window. Enter your search criteria in the Find field.

❖ **Printing performance data**

1 Select Tools | Generate Report.

2 Select the level of details that you want to print and choose the destination file. Available level of details include:

- Overview
- Job Performance (appears only when you are printing performance data reports for jobs)
- Project Performance
- Component Performance

3 Enter a destination file path for the generated report, or accept the default value.

4 To limit the report data to the executions selected in the Overview tab, select “Only selected executions.”

Note “Only selected executions” is not available if you do not select an execution in the Overview tab.

5 Click Generate. On successful generation of the report, a message displays. Click Yes to view the performance data report.

The report displays tabular and graphical data. It includes a table of contents that provides links to corresponding sections within the report.

The Performance Overview section of the report displays the duration of each project or job execution. The Project, Component, and the Job performance sections display a table and a chart of performance data for the selected execution, each showing a different level of data granularity.

Performance data model and content

Performance data is stored in a single, denormalized, repository table named TRON_PERFORMANCE, which you can query to collect performance data.

Each execution of project or job is identified by a global unique ID. You will find the execution starting time in three variations: a full timestamp, a date, and a time. Additional information is provided about the account that initiated the execution and the repository in which the project or job is located.

Note The starting time for execution and events are logged in Coordinated Universal Time (UTC), also referred to as Greenwich Mean Time (GMT).

For each executed project ID, version (modification date), name, and a global, unique execution ID is reported. A single project execution event stores the duration of a project in ms.

Project components are represented by ID, name, class, type, and version. The process event provides the number of steps and the amount of processed records.

For port events, the ID, name, class, type and the amount of input or output blocks and records are reported.

For each job, the ID, the version (modification date), and the name is reported. A single job execution event stores the duration of a job in ms. The components (projects) of a job are represented by their ID, class, and version.

Events

The performance log is based on events. For each event, the starting time (in three variations: a full timestamp, a date, and a time) and the duration (in ms) is stored. The description of an event is made up by a class, a name, and a text. Some events include a result (such as succeeded or failed). Also included is information about the engine that reported an event.

Reported events include:

Class	Name	Description
control	execute job	Total execution time of a job (one record per job execution, job duration in ms, in attribute PRF_JOB_DURATION).
init	load job	Get job definition from repository.
control	execute project	Total execution time of a project (one record per project execution, project duration in ms, in attribute PRF_PRJ_DURATION).
init	load project	Get project definition from repository.
init	create	Create project and component instances.
init	configure	Configure project and component instances.
perform	prepare	Perform component preprocessing.
perform	process	Perform component step.
perform	finish	Perform component postprocessing.
perform	read	Get data to component IN-port.
perform	write	Push data from component OUT-port.

Note Due to distributed multithreading, the total project execution time can be significantly shorter than the sum of the execution time of all participating components.

Configuring alerts for runtime events

You can set alerts to be notified by e-mail messaging at the occurance of runtime events such as, when projects or jobs start, complete, or generate an error message. To set these alerts on the engine you are connected to, select Tools | Alert Manager from the ETL Development UI. The Alert and Event Configuration window appears:

- Engine pane – select the engine to configure the alert on. By default, the selected engine is the one started by ETL Development.
- Events pane – includes a navigation tree that displays different event types:
  - Jobs – occurs when a job starts, finishes normally, or finishes with errors.

- Job Start – occurs when a job starts.
- Job Finish – occurs when a job finishes normally.
- Job Error – occurs when a job finishes with errors.
- Projects – occurs when a project starts, finishes normally, or finishes with errors.
- Project Start – occurs when a project starts.
- Project Finish – occurs when a project finishes normally.
- Project Error – occurs when a project finishes with errors.

- Alerts pane – displays event to alert mappings.
- Alert Definition pane – used for configuring alerts.

❖ **Creating an alert for an event**

1 Select the engine on which you want to configure the alert.

2 Select the event type.

3 In the Alerts pane, click the Add icon.

4 Provide a unique name for the new alert and click OK.

5 In the Alert Definition pane, specify e-mail preferences and filter conditions for the new alert.

- E-mail Preferences:

  a Enter appropriate values in the “To,” “CC,” and “BCC” fields.

  b You can manually enter the content in the subject or the body field, or click the “Select Event Property” icon to select event properties. For example, if you select the [Job Name] event property, when the alert is triggered, the event property is replaced with “Job 1,” which is your real job name.

  c Click Test Mail to test the e-mail preferences.

  Note The content of the subject and body in the test mail is generated automatically by the system.

- Filter:

  a Click the “Select Project or Job” icon and choose the project or job that should invoke the alert. Click OK.

The filter condition is generated automatically. However, if required, you can manually edit the filter condition.

If you are entering the filter condition manually, click the "Select Event Properties" icon to choose the event properties.

b Select Include to be notified when the events for the project or job match the filter condition. Select Exclude to be notified when the filter conditions are not met.

c Select "Enable alert during simulation" to be notified of an event during simulation. This option is available only for project events.

Click Save.

❖ **Editing an alert**

1 In the Alerts pane, click the alert.

2 Modify the alert preferences and click Save.

❖ **Deleting an alert**

- Select the alert to remove and click the Delete icon.

❖ **Copying an alert (Save As)**

1 Select the alert you want to copy.

2 Click the Save As icon.

3 Enter a name for the new alert.

4 Modify the e-mail preferences and filter conditions for the alert, if required. Click Save.

Note Sybase recommends that you set up e-mail alert notifications on one machine. You can then manually copy the alert configuration file across different grid engines running on the network. Use a text editor to define the alerting configuration in the "SMTP" section of the *default.ini* file in the *etc* directory.

CHAPTER 5

Components

This chapter provides detailed descriptions of various Sybase ETL components.

Overview

Sybase ETL components are used to create projects and jobs. These components are available in the Component Store. Project components include:

- Source components – deliver data for a transformation stream. A project typically starts with one or more source components. Source components have no IN-ports and at least one OUT-port.
- Transformation components, Lookup components, and Staging components – apply specific transformations to the data in the transformation stream. These types of components have both IN-ports and OUT-ports.
- Destination components (also called data sinks) – write data to specific targets. destination component have one IN-port and no OUT-ports.
- Loader components – extract and load data from a source database or file into the IQ database, without performing any transformation.

Setting up component properties

Each component is dedicated to a specific task, and incorporates task-specific features. However, you can use the same procedure to set up properties for all component types.

Before you can use a component in a project, you must set the required properties, using either:

- The Configuration window that appears when you add a component to the Design window, or,
- The Property window for a component added to the project.

Evaluating SBN expressions

Set the Evaluate property for Square Bracket Notation (SBN) expressions to be evaluated before a property value is used. See “Square Bracket Notation” on page 63.

For some properties, the Evaluate property is selected by default.

You can change the Evaluate properties in either of these ways:

- In the Properties window, select the Eval option for the property or,
- Right-click a property and select Evaluate.

Note If you select the Eval option for the Password property, the value appears as plain text.

Encrypting properties

Property values are stored in the Sybase ETL repository. However, most entries in the repository are not encrypted, and are represented as readable character sets.

To toggle encryption properties, select the Encrypt option in the Properties window, or right-click the property and select Encrypt.

Property reference variable

Properties with simple values have an associated variable that you can reference in component expressions and procedures. The variable always contains the current value of a property evaluated, if Evaluate is set.

- To display the variable name, right-click the property in the Properties window and select Reference Variable.
- When setting up transformation rules in the JavaScript Debugger, click Variable | Parameter to access the Reference Variables.

Custom properties

You can add custom properties to components. Custom properties incorporate variables that you can reference in component expressions or procedures. They can contain expressions that are assigned in parameter sets or that are evaluated at runtime.

❖ **Adding custom properties**

1 In the Design window, select the component.

2 In the Property window, right-click and select Add.

3 Enter the name for the property. Do not use reserved JavaScript keyword. See “Variables” on page 62. Inside the component, this property is referenced using the variable REF.*<name of property>*.

4 In the Prompt field, enter a value to appear as the label for this property in the Properties window.

5 In the Description field, enter a description to be used in the property tooltip.

6 Click OK.

❖ **Removing custom properties**

1 In the Property window, right-click the custom property and select Remove.

2 Click OK to confirm.

Note You must also remove all references to the associated variables.

Providing descriptions to components

You can assign a name and a description to a component. The description and the name appear in the tooltip, which displays when you move your mouse over a component. The name also appears at the top of the component.

❖ **Adding description to components**

1 In the Design window, right-click the component and select Description.

2 Enter a name and description for the component. You can use HTML formatting tags to format the description.

Configuring port structure

A component has IN-ports, which receive data, and OUT-ports, which pass the data after processing. When you step through a component, the data forwarded through the OUT-port is delivered to the IN-port of the next component.

Managing port structures

An unstructured port inherits the structure of a structured port, when a connection is added between them.

You can add attributes to a port structure.

After you add a component to the project, the color of the component ports indicates the status of the component:

- Green – component is properly configured.
- Yellow – the port structure is defined but one or more mandatory properties are not defined.
- Red – no port structure is defined, but one or more mandatory properties are not defined.

Note To enhance support for users with color disabilities, you can change the color of the yellow component port by selecting “Use enhanced color accessibility” in the File | Preference window. See “Customizing preferences” on page 19.

❖ **Modifying port structures**

1 In the Design window, right-click the port and select Edit Structure.

2 In the Structure Viewer window, make the required changes, and click Save.

Note If a port structure cannot be modified, the Edit Structure option is not available. You can only view the port structure using the View Structure option.

❖ **Adding port attributes**

1 In the Design window, right-click the port and select Edit Structure.

2 In the Structure Viewer window, select Actions | Add, or right-click the attribute and select Add.

3 Enter a name for the new attribute. The names for port attributes must start with an alphabet character and can contain only alphanumeric characters. Do not use a reserved JavaScript keyword. See “Variables” on page 62.

4 Select the Populate Attribute option to add the attribute to multiple port structures. The new attribute is added to all port structures participating in the selected connections and is automatically mapped. Click OK.

❖ **Deleting port attributes**

1 In the Design window, right-click the port, and select Edit Structure.

2 In the Structure Viewer window, select Actions | Remove, or right-click the attribute and select Remove.

❖ **Modifying port attributes**

1 In the Design window, right-click the port and select Edit Structure.

2 In the Structure Viewer window, change the attribute settings, and click Save.

❖ **Copying port structures from other ports**

You can assign a port structure to a port based on any other available port in the current project:

1 In the Design window, select the port you want to assign a new structure to.

2 Right-click and select Assign Structure | Copy Structure.

You can copy the port structure from other ports of the same component. You can also copy any other port structure of the current project by selecting Copy Structure.

If you select Copy Structure, a window displays an overview graph of the current project. You can select any of the available ports in the project.

❖ **Updating port structure with database changes**

To apply database changes to the port structure of the DB Source, DB Sink, and DB Staging components:.

- Right-click the component and select Reconfigure.

The Reconfigure option updates the component configuration when there is a change in the database schema. It closes the current connection, opens a new connection to the database, reads the metadata of the query, and applies the updates to the port structure.

❖ **Selecting a port to serve as the source for your new port structure**

1. Click the port you want to use as a source. The attribute structure of the selected port appears in the lower area of the window.
2. Click Apply.

Simulating components

Initializing components

In the Design window, right-click the component, and select Initialize, or Initialize and Step.

If you modify one of the existing property settings of a component during simulation, reinitialize the component before moving forward.

Stepping a component multiple times

You can step a component multiple times during a simulation, to preview the behavior of a component with different property settings. Repetitive stepping does not increase the number of records delivered to the downstream component.

❖ **Stepping through components multiple times**

1. In the Design window, click the component.
2. In the Properties window, modify the transformation rules or property settings.

3 Right-click the component and select Initialize.

4 Right-click the component and select Step.

5 Return to step 2.

When stepping a component repetitively, the same set of records at the IN-port are reprocessed in each step and forwarded without increasing the number of the records at the OUT-port. You can step through many components from inside the component window, or by using the menu that appears when you right-click in the Sybase ETL Development window.

Previewing transformation results

When working in simulation mode, you can divide the entire result set of the data source (as defined by a query in the source or staging component) into data blocks, which contains a subset of records. The number of records in each subset is related to the Read Block Size parameter, which appears in the Property window of the component. To enhance performance when stepping through the project, select a small number for the Read Block Size parameter.

❖ **Previewing transformation results**

1 Step through the project, including the component you want to preview.

2 Right-click the component, port, or connecting link, and select Preview.

If a component has multiple ports, first select the port for which you are previewing transformation results.

Database connection settings

Specify database connection parameters in the:

- Database Configuration window, which appears when you add a component to the Design window, or when you double-click an existing project component.
- Properties window, which appears when you select a project component in the Design window.

The database connection parameters are:

- Interface – select the method or driver you want to use to connect to the destination or source database. To set special database options, click the Database options icon.

  Note Sybase Open Client™ must be installed on the same computer as Sybase ETL Development, and the database server must be defined in the *%SYBASE%\ini\sql.ini* file on Windows, and in the *$SYBASE/interfaces* file on UNIX and Linux. If you are executing a project on an ETL Server, the ETL Server must also have access to Open Client libraries.

  The available interfaces for a component can be one or more of:

  - Sybase
  - ODBC – the ODBC driver must be installed on the same computer as Sybase ETL Development, and a system data source name (DSN) must be defined for the target. If the project is to be executed on an ETL Server, the ETL Server must also have access to the proper ODBC drivers and DSN.
  - OLE DB
  - Oracle
  - DB2
  - SQLite Persistent

- Host Name – select the source or destination database. The options that appear on the Host Name list depend on the interface you selected.

  Note If you selected the SQLite Persistent interface, you can enter an existing or new database file name. See "Connecting to a SQLite database" on page 305.

- Database user name and password – specify the authorized database user name and password.
- Database name – specify the database you want to use as the source or destination database.
- Database schema – specify schema or owner to restrict the objects displayed and create new tables in that schema.
- Standardize Data Formats – convert incoming date and number information into a standard format, which Sybase ETL can move between systems that support different formats.
- Database options – override performance defaults and control the behavior of some transactions. For a list of database options, see "Interface-specific database options" on page 299.

Source components

Source components deliver data for a transformation stream. A project typically starts with one or more source components. Source components have no IN-ports and at least one OUT-port.

Component	Description
DB Data Provider Full Load	Extracts data from any data source accessible by an ODBC connection or a native driver (DB2, Oracle, Sybase), or from Microsoft SQL Server via OLE DB.
DB Data Provider Index Load	Performs incremental data loads. The incremental load is controlled by an ascending index attribute that contains the ascending values.
Text Data Provider	Reads and transforms structured data from a text file into a table.
XML via SQL Data Provider	Loads hierarchical XML data into a relational schema.

Component	Description
CDC Provider Sybase Replication Server	Receives source table data changes from Replication Server, translates the data to the standard ETL data flow, and then sends the data to the next component.

DB Data Provider Full Load

DB Data Provider Full Load is a Source component that extracts data from any data source that is accessible by an ODBC connection or native driver (DB2, Oracle, Sybase), or from Microsoft SQL Server via OLE DB. The structure of the single OUT-port mirrors the structure of the query result set.

Configuring a DB DataProvider Full Load component

1 Drag the DB DataProvider Full Load component onto the Design window. The Database Configuration window appears. Alternatively, to open the configuration window, select the component in the Properties window, and click the Properties icon.

2 Enter the connection parameters. See “Data Provider Full Load properties list” on page 96. You must add a valid interface and host name.

3 Click the Query icon. Create and save a query to retrieve the data set from the data source.

You can create a query directly in the Query window, or click the Query Designer icon to open Query Designer and generate queries. See “Query Designer” on page 51.

4 Click Finish.

5 In the Properties window, specify any other optional properties and database options.

See “Managing port structures” on page 90 for information on how to modify port structures and port attributes.

Data Provider Full Load properties list

DataProvider Full Load properties list identifies the connection parameters and other items you must define in the Property window.

Required properties

Property	Description
Interface	Identify the method or driver to use to connect to the data source.
Host Name	Identify the data source. The options that appear in the Host Name list depend on the interface you select.
Query	Create a query that retrieves information from the data source. Use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer. See "Query Designer" on page 51.

Optional properties

Property	Description
User and Password	Identify an authorized database user and protect the database against unauthorized access.
Read Block Size	Determine the number of records retrieved by the component in a single step.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema or owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in the schema you specify.

Property	Description
Standardize Data Format	Convert incoming date and number information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (“.”) as the decimal separator.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Transactional	All work performed by the DB Data Provider Full Load component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

DB Data Provider Full Load demos

Sybase ETL includes several demonstrations for the DB Data Provider Full Load component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Source | DB Data Provider - Full Load.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer. Repository | Projects. Then select:

- Demo Transfer German Customers
- Demo Transfer German Products
- Demo Transfer German Sales
- Demo Transfer U.S. Customers
- Demo Transfer U.S. Products

DB Data Provider Index Load

DB Data Provider Index Load is a source component that performs incremental data loads based on an ascending index value. During execution, DB Data Provider Index Load ignores any previously extracted data records.

Use DB Data Provider Index Load to perform incremental loads that track source changes on a regular basis.

Impact on the Simulation sequence:

- The Read Block Size value impacts the number of records loaded in a single simulation step.
- The value of the Load Index is not updated in the repository when the project is executed during simulation.

Configuring a DB DataProvider Index Load component

1. Drag the DB DataProvider Index Load component onto the Design window. The Database Configuration window appears. Alternatively, to open the configuration window, select the component in the Properties window, and click the Properties icon.
2. Add the connection parameters. See the “DB Data Provider Index Load properties list” on page 100 for specific field requirements. You must add a valid interface and host name.
3. Click Finish.
4. In the Properties window, select an ascending index attribute from the list of database objects.
5. Click the Query icon. Create and save a query to retrieve the data set from the data source.

   You can create a simple query directly in the Query window, or click the Query Designer icon. See “Query Designer” on page 51.
6. In the Properties window, specify any other optional properties and database options.

❖ **Resetting the ascending index value**

You cannot directly manipulate the persistent value of load index in the repository. However, you can reset the value to the one stored in the Load Index Value property. To reset execution properties:

- Right-click the project in the Navigator and select Reset Execution Properties.

❖ **Updating port structure with database changes**

- See “Updating port structure with database changes” on page 92.

DB Data Provider Index Load properties list

DB Data Provider Index Load properties list identifies the connection parameters and other items you must define in the Database Configuration window.

Required properties

Property	Description
Interface	Identify the method or driver you want to use to connect to the data source.
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.

Query	Create a query that retrieves information from the data source. Use the predefined variable *LoadIndex* to qualify the selection criteria in the WHERE clause. Enclose *LoadIndex* with square brackets, because it is evaluated before the query is sent to the database, for example:
	`select * FROM SALES`
	`WHERE SA_DELIVERYDATE >'[LoadIndex]'`
	`ORDER BY SA_DELIVERYDATE`
	Note Quote characters differ between database systems. In Microsoft Access databases, use # for datetime values.
	Use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer. See “Query Designer” on page 51.

Ascending Index	Select the attribute that contains the ascending index for delta load.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.

Load Index Value	The maximum value of the ascending index attribute is automatically used and stored when executing a Sybase ETL job or schedule. The Load Index Value simulates the project using the specific value provided by the user. For example:
	`2005-01-19`
	`100`

Read Block Size	Determine the number of records retrieved by the component in a single step.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Convert incoming date and number information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (“ . ”) as the decimal separator.

Property	Description
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Transactional	All work performed by the DB Data Provider Index Load component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

Simulating an Index Load

When a project containing an Index Load component is fully configured you can simulate the incremental load:

1 To run the project, select Run | Trace. All records matching the condition specified by the Load Index Value property are processed.

2 Select to perform post-processing for a successful execution. Click Yes.

3 Simulate the project or the Index Load component again. The Index Load component does not return any records. See “Simulating a project” on page 27.

Note You can manually update the source table between step 1 and 2 to verify that the correct modified records are retrieved.

4 To simulate again, right-click the component and select Reset Load Index Value, or, in the Properties window, enter a new value for the Load Index Value property.

Data Provider Index Load demos

Sybase ETL includes a demonstration for the Data Provider Index Load component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Source | DB Data Provider – Index Load.

To access the sample project, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select Demo Transfer U.S. Sales on an incremental basis.

Text Data Provider

Text Data Provider reads and transforms structured data from a text file into a table. The text source must contain fixed-length or delimited fields.

Configuring a Text Data Provider component

1 Drag Text Data Provider to the Design window.

2 In the Text Data Provider component window, select the text file to use as a data source.

The Text Data Provider component window lets you define the structural properties of data at the OUT-port. It includes:

- File Content pane – displays the contents of the source document.
- Properties pane – the file description properties. You can modify the file description properties, if required. See “Text Data Provider properties list” on page 105 for specific field requirements.
- Output Port Content pane – a tabular view of data at the OUT-port. In the Output Port Content pane, click the “Regenerate the column definition icon to regenerate column definitions. If you want the column names to be read from a data file, click the “Read column names from a data file” icon. See “Reading column names from a data file” on page 103.

Reading column names from a data file

1 In the Output Port Content pane of the Text Data Provider component window, click the “Read column names from the data file” icon.

2 Provide the line number of the record that contains the column heading. Click Enter to confirm.

You can double click the column headings to edit the name of the column.

Skipping row delimiters

The line containing the column headings is not automatically skipped when the data is processed. To skip a specified number of rows at the beginning of the input table(s) for a load, enter a value in the “Skip First Rows” field.

If the rows to be skipped do not contain the same format as later rows; for example, if there is a “header” row with no column delimiters, while later rows all contain 5 columns, it is still counted as 1 row.

Note Quoted row delimiters are not regarded as row delimiters.

Delimiter considerations

This section discusses the considerations to keep in mind when using quote characters, row and, column delimiters.

A value can have quote characters if:

- Quotes start at the beginning of the value
- Quotes surround the entire value

If a value starts with a quote, everything up to the next quote delimiter combination is read, else everything up to the next delimiter is read. For example, if Column A is a single column of data, it can be quoted as “Column A”. However, quoting values in any of these ways is invalid:

- “Column” A – quotes don’t surround the entire value.
- “Column “A” – quotes don’t start at the beginning of the value.

If you have 2 columns of data, say “Column A” and “Column B,” which are separated by a comma Column Delimiter and a CRLF Row Delimiter:

- The value can be quoted as “Column A”, “Column B” <CRLF>.
- The value cannot be quoted as “Column” A, “Column B” <CRLF>.

Here are some examples of using quote characters and delimiters:

If you are using:

- Double quotes (“) as Column Quotes
- Comma (,) as the Column Delimiter
- Line feed (<LF>) as the Row Delimiter

If the value from the source file is:

- “ABCD”, “DEF” <LF> – Column 1 will be read as ABCD and Column 2 will be read as DEF.
- ““A””, “D,E,F” <LF> – Column 1 will be read as “A” and Column 2 will be read as D,E,F.
- ““A”, “D,E”<LF> – Column 1 will be read as “A and Column 2 will be read as D,E.

Text Data Provider properties list

Text Data Provider properties list identifies items about the structure of source files. Properties are initially set when you add the component to the project.

Required properties

Property	Description
Text Source	Identify the text file to use as the data source. You can select the data source when you add Text Data Provider to a project, or from the Properties window. To select a data source from the Properties window, click the Text Source icon, then select the file.
Columns	Define columns for the data in your source file.

Optional properties

Property	Description
Row Delimiter	Specify how each row is delimited: • Position (fixed line position) • LF (line feed) • CR (carriage return) • CRLF (carriage return followed by a line feed) Alternatively, you can enter a different delimiter.
Row Length	Specify the number of characters in each fixed row, if you have selected Position as the Row Delimiter.
Column Delimiter	Specify how columns are delimited: • Position (fixed column positions) • Tab • Comma • Semicolon Alternatively, you can enter a different delimiter.

Property	Description
Column Quote	Specify how the values in the source file are quoted: • None • Single quote • Double quote Alternatively, enter a different quote character or string.
Fixed by Bytes	Specify how to interpret the values provided for line length, column start, and column end: • Not selected (default) – values are interpreted as number of characters. • Selected – the values are interpreted as number of bytes. For example, suppose your source file includes **abcÖÐÎÄ abcdef** and has these characteristics: • File Type – Fixed (Variable Line) • Encoding – GB2312 • Row delimiter – '\n' • Column definition – column1: 1-7; column 2: 9-10 If you select the Fixed by Bytes option: • Column 1 displays the first 7 bytes, **abcÖÐÎÄ**. • Column 2 displays the 9th and 10th bytes, **ab**. If you do not select the Fixed by Bytes option: • Column 1 displays the first 7 characters, **abcÖÐÎÄ a**. • Column 2 displays the next 2 characters, **cd**.
Null Byte Substitute	Set the character to replace null bytes.
Skip Rows	Skip a specified number of rows in the row sequence.
Encoding	Set the current character encoding.
Support Unicode	Set Unicode support.
Read Block Size	Determine the number of records retrieved by the component in a single step.
Read empty as NULL	Replace values read as “empty” with “null.”

Text Data Provider demos

Sybase ETL includes several demonstrations for the Text Data Provider component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Source | Text Data Provider.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select:

- Demo Transfer German Customers
- Demo Transfer German Sales
- Demo Transfer U.S. Customers

XML via SQL Data Provider

The XML via SQL Data Provider component loads hierarchical XML data into a relational schema that you can query like a relational database.

XML via SQL Data Provider is designed for data-centric XML documents, such as sales order, stock quotes, or scientific data, which are characterized by a regular hierarchical structure.

Configuring an XML via SQL Data Provider component

1 Drag the XML via SQL Data Provider component onto the Design window.

2 In the Properties window, click the XML Source icon and select the XML to use as a data source. You can specify *HTTP*, *FTP*, *URL*, or file name.

3 Click the Data Output icon.

4 Click the Properties icon to open the XML Port Manager. Specify a query for each OUT-port.

By default, XML via SQL Data includes one OUT-port, but you can add ports. Any OUT-ports you add in XML Port Manager appear in the Design window. See “Working with XML Port Manager” on page 108.

❖ **Updating the OUT-port structure**

To update the OUT-port structure to reflect changes made to the XML source file:

- Right-click the XML via SQL Data Provider component and select Reconfigure.

  The Reconfigure option updates the component configuration when there is a change in the database schema. It closes the current connection, opens a new connection to the database, reads the XML source file, and applies the updates to the output port structure.

Working with XML Port Manager

The XML Port Manager window lets you create queries against the XML source file, to define one or more output data streams.

- XML source view – displays the contents of the source document.
- Data Model tab – displays a relational view of the source document.
- Reference tab – displays the available component variables.
- Output port area – used to write queries against the data model, and send the results to a particular OUT-port. Although XML Port Manager is by default, configured with one OUT-port, you can add additional ports, and write additional queries.

Writing queries

When you open XML Port Manager, the OUT-port area includes a standard query against the XML view, which returns all columns and all rows to OUT1. Assume that your XML source document contains customer data expressed as attribute values of each data node:

```
<root>
    <data id="101" fname="Michaels" lname="Devlin"
       address="114 Pioneer Avenue" city="Kingston"
       state="NJ" zip="07070"/>
    <data id="102" fname="Beth" lname="Reiser"
       address="33 Whippany Road" city="Rockwood"
       state="NY" zip="10154"/>
     <data id="103" fname="Erin" lname="Niedringhaus"
       address="190 Windsor Street" city="Tara"
       state="PA" zip="19301"/>
</root>
```

To retrieve customer attributes against the XML, you can use a query similar to:

```
select * from V_XML_CONTENT WHERE TAB_data_ATT_city =
'Kingston'
```

This query opens the Content Browser, and returns only those rows whose city value matches Kingston. In the tabular view, you can write a query that looks similar to:

```
select * from TAB_data where ATT_city='Kingston'
```

Note XML data relationships are generally expressed as parent/child, or as node/attribute relationships. The Content Browser returns XML Port Manager query results as columns and rows. Column and row references refer to query results, not XML data.

❖ **Retrieving data from your XML data source**

- Use standard SQL syntax to write your queries directly into the port field in the OUT-port area:

```
select column
FROM table_name
```

- Query Designer helps you design queries for the tabular view. Depending on the structure of the XML source, you may need to create joins between the tables to return rows of data.
- The default XML view_name is `V_XML_CONTENT` . To return a row, qualify the select statement with a WHERE clause (`select * from V_XML_CONTENT WHERE TAB_state_ATT_state = 'NY'`).
- In the tabular view, XML nodes formatted as attribute expressions sometimes create a wrapper element you can qualify with a WHERE clause (`select * from TAB_data where ATT_city='Kingston'`) to return a row.

❖ **Writing queries against the Table view**

- You can write queries against the Table view directly into the port field in the OUT-port area, or use the Query Designer. Use standard SQL syntax to query tables in the Table view.

❖ **Adding and removing ports**

- Right-click the port section and select Add Port or Remove Port.

Setting up a sample project

This section guides you through setting up the XML via SQL Data Provider component using a simple example. To follow this example, use the *PRODUCTS.xml* as the XML source; it is located in the Demodata subdirectory of the Sybase ETL installation directory.

XML Port Manager

Open XML port manager, and define the ports in the OUT-port area. Each port is described by a select statement based on the XML Data Model tables.

XML source

The following XML document is a simple product structure. Each product is described with an ID (PR_ID), name (PR_NAME), product group (PR_GROUP1), and price (PR_PRICE), for example:

```
<?xml version="1.0" encoding="UTF-8"?>
    <dataroot xmlns:od="urn:schemas-solonde-com:demodata"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="PRODUCTS.xsd"
            generated="2005-01-24T16:13:26"><PRODUCTS>
        <PR_ID>435672</PR_ID>
        <PR_NAME>24 CD Rom Drive</PR_NAME>
        <PR_GROUP1>CD Rom</PR_GROUP1>
        <PR_PRICE>134</PR_PRICE>
    </PRODUCTS>
    <PRODUCTS>
        <PR_ID>435673</PR_ID>
        <PR_NAME>Notebook 235</PR_NAME>
        <PR_GROUP1>Notebook</PR_GROUP1>
        <PR_PRICE>1455</PR_PRICE>
    </PRODUCTS>
</dataroot>
```

The data model

There is one table for the root element (TAB_dataroot), followed by one or more tables for elements at level 1. In the example, only one element at this level exists (TAB_PRODUCTS). At the next level, you find a table for each element on level 2 (TAB_PR_ID, TAB_PR_NAME, TAB_PR_GROUP1, TAB_PR_PRICE). There can be more nested levels in the XML document, and each level creates another set of tables.

Note You can change the prefixes for the generated table names in the DB Schema Options property.

Root level	Elements level 1	Elements level 2
TAB_dataroot ATT_ROW_ID ATT_FK_generated ATT_xmlns_od ATT_xsi_no	TAB_PRODUCTS ATT_ROW_ID ATT_FK_dataroot	TAB_PR_ID ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_ID TAB_PR_NAME ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_NAME TAB_PR_GROUP1 ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_GROUP1 TAB_PR_PRICE ATT_ROW_ID ATT_FK_PRODUCTS ATT_PR_PRICE

The tables are linked through foreign keys. Table TAB_PRODUCTS is linked to TAB_dataroot through the ATT_FK_dataroot attribute.

The tables at level 2 are linked to table PRODUCTS through the ATT_FK_PRODUCTS attribute. To create the view containing the PRODUCTS records, the tables at level 2 must be joined with the TAB_PRODUCTS table.

The join is qualified by the ATT_FK_PRODUCTS attribute of each level 2 table and the ATT_ROW_ID of TAB_PRODUCTS. The only selected attributes are the value attributes of level 2 tables: ATT_PR_ID, ATT_PR_NAME, ATT_PR_GROUP1 and ATT_PR_PRICE.

```
select  TAB_PR_ID.ATT_PR_ID,
TAB_PR_NAME.ATT_PR_NAME, TAB_PR_GROUP1.ATT_PR_GROUP1,
TAB_PR_PRICE.ATT_PR_PRICE
FROM  TAB_PRODUCTS, TAB_PR_ID, TAB_PR_NAME,
TAB_PR_GROUP1, TAB_PR_PRICE
WHERE  TAB_PR_ID.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID AND
TAB_PR_NAME.ATT_FK_PRODUCTS = TAB_PRODUCTS.ATT_ROW_ID
AND  TAB_PR_GROUP1.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID AND
TAB_PR_PRICE.ATT_FK_PRODUCTS =
TAB_PRODUCTS.ATT_ROW_ID
```

XML via SQL Data Provider properties list

XML via SQL Data Provider Properties List sets processing options for the XML source file.

Required properties

Property	Description
XML Source	Identify the data source. You can select the XML data source when you add a component to a project, or select the file from the Properties window. To select a data source from the Properties window, click *XML Source*, then select the file.
Data Output	Open the XML port manager, a management console where you can query XML source. See “Working with XML Port Manager” on page 108.

Optional properties

Property	Description
Document Schema	Identify an external schema (*.xsd*) or DTD that you can use to validate the XML source.
Namespace Schema	Point to the location of an external namespace schema. An XML schema consists of components such as type definitions and element declarations that can be used to assess the validity of well-formed element and attribute information items.
Validate Schema	Enable schema and DTD validation.

Property	Description
XML Options	Set these XML processing options: • Full schema check – set this item to 1 to check for items that may be time consuming or memory intensive. Particle unique attribution constraint checking and particle derivation restriction checking are controlled by this option. The default value is 0. • Ignore external DTD – set this value to 1 to ignore an external DTD referenced within the document. The default value is 0. • Process namespace – set this value to 0 if you do not want namespace specification to be considered during parsing. The default value is 1. • Preserve Element whitespace – set this value to 1 to preserve white space in the XML element value. Set to 0 to trim white space from the XML element values. If the XML element value includes only white space and the value is set to 0, it is interpreted as an empty element value.
DB Schema	Select the database schema setup (create tables) script. Use this option to enforce a fixed data model.

Property	Description
DB Schema Options	Customize the settings for tables and attributes generated from the XML structure, including the prefixes for table and attribute names. The DB Schema options are: • Attribute name case – formats the attribute names generated from the XML. The values “upper” and “lower” convert attributes names accordingly. “Mixed” (the default) leaves the names as they appear in the XML document. • Attribute name prefix – prefix to be used for every generated attribute name. • Create indexes – set to1 (the default) to automatically generate indexes on the primary keys of the tables. • Create flat views – set to1 (the default) to automatically generate a view called V_XML_CONTENT. This view joins all tables and returns all XML data in a broad table. If the database schema results in more than 32 tables, this view does not work and the option has to be switched off. • Foreign key prefix – prefix to be used for attributes that are foreign keys. • Ignore Empty Leaf Element Values – set this value to 1 if you do not want the database to include column entries for specific XML leaf elements that contain no data. If set to 0, the database created from the XML document includes column entries for all XML leaf elements whether empty or not. • Primary key name – attribute name to be used for primary keys. • Table name case– formats the table names generated from the xml. The values “upper” and “lower” convert the table names accordingly. “Mixed” leaves the names as they appear in the XML document. • Table name prefix – prefix to be used for generated table names
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Read Block Size	Determine the number of records retrieved by the component in a single step. If the component has more than one OUT-port, the read block size is ignored and the component provides data at all ports, in a single step.

XML via SQL Data Provider demos

Sybase ETL includes a demonstration for the XML via SQL Data Provider component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Source | XML via SQL - Data Provider.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select Demo XML via SQL Data Provider.

CDC Provider Sybase Replication Server

The CDC (Capture Data Changes) Provider Sybase Replication Server component is used for implementing incremental load. It:

- Receives data changes from Sybase Replication Server, translates them to an ETL standard data flow, and then sends the data out to the next component.
- Automates the process of configuring Sybase Replication Server, which includes marking the source table as replicated, creating or dropping replication definitions, connections, and subscriptions. This entire process is termed as Creating or Dropping Replication, and it enables Replication Server to start or stop capturing source table data changes.

Note CDC Provider Sybase Replication Server component only supports ASE and Oracle as the source database.

Before configuring the CDC Provider Sybase Replication Server component:

- Install Replication Server.

  Note You need not install Replication Server and ETL Server on the same machine.

- Allocate sufficient disk space for the Replication Server partition. See Chapter 3, “Replication Server Commands,” in the *Replication Server 15.2 Reference Manual*.
- Configure the Replication CDC Service:

a Edit the *interfaces* files for the source database, Replication Server, and ETL Server. See "Updating the interfaces files for the source database, Replication Server, and ETL Server" on page 116.

b Configure the Replication CDC Service Name for each ETL Server. See "Configuring Replication CDC Service name for each ETL Server" on page 118.

- If the source database is Adaptive Server® Enterprise, use rs_init to add the source database to the Replication Server. See the *Replication Server Configuration Guide*.
- If the source database is Oracle, see "Configuring Oracle as a replication source" on page 119, on how to add Oracle to the Replication Server.

Updating the *interfaces* files for the source database, Replication Server, and ETL Server

1 Navigate to the *interfaces* file.

- On Windows, the file is in <*installation_directory*>*ini**sql.ini*.
- On UNIX and Linux, the file is in <*installation_directory*>*/interfaces*

2 Use a text editor to modify all the three *interfaces* files to include:

- Entries for Replication CDC Services of all ETL Servers.
  - On Windows for example, use:

```
[<cdc_service_name>]
master=TCP,<machine_name>,<port>
query=TCP,<machine_name>,<port>
```

  - On UNIX or Linux, use:

```
<cdc_service_name>
master tcp sun-ether <machine_name> <port>
query tcp sun-ether <machine_name> <port>
```

  where:

  <*cdc_service_name*> is a unique Replication CDC Service name to be used by a single ETL Server.

  <*machine_name*> is the name of the machine on which the grid ETL Server runs.

  <*port*> is the port on which the Replication CDC Service listens.

- A SYBETL_VIR_RDBMS entry that contains all CDC Service entries.
  - On Windows for example, use:

```
[SYBETL_VIR_RDBMS]
master=TCP,<machine_name_1>,<port>
query=TCP,<machine_name_1>,<port>
master=TCP,<machine_name_2>,<port>
query=TCP,<machine_name_2>,<port>
```

  - On UNIX and Linux, use:

```
SYBETL_VIR_RDBMS
master tcp sun-ether <machine_name_1> <port>
query tcp sun-ether <machine_name_1> <port>
master tcp sun-ether <machine_name_2> <port>
query tcp sun-ether <machine_name_2> <port>
```

Note The IP port of the virtual database and Replication CDC Services should match.

- If the source database is ASE, create or copy the entries for your ASE source database, Replication Server, and Embedded Replication Server System Database (ERSSD) or RSSD.

  If the source database is Oracle, create or copy the entries for your Replication Agent™, Replication Server, and Embedded Replication Server System Database (ERSSD) or RSSD.

  - On Windows for example, use:

```
[<ase_name>]
master=tcp,<ase_machine_name>,<ase_port>
query=tcp,<ase_machine_name>,<ase_port>

[<repserver_name>]
master=tcp,<repserver_machine_name>,
       <repserver_port>
query=tcp,<repserver_machine_name>,
       <repserver_port>


[<erssd_name>]
master=tcp,<erssd_machine_name>,<erssd_port>
query=tcp,<erssd_machine_name>,<erssd_port>
```

  - On UNIX and Linux, use:

```
<ase_name>
master tcp sun-ether <ase_machine_name>
<ase_port>
query tcp sun-ether <ase_machine_name>
<ase_port>

<repserver_name>
master tcp sun-ether <repserver_machine_name>
<repserver_port>
query tcp sun-ether <repserver_machine_name>
<repserver_port>


<erssd_name>
master tcp sun-ether <erssd_machine_name>
<erssd_port>
query tcp sun-ether <erssd_machine_name>
<erssd_port>
```

Configuring Replication CDC Service name for each ETL Server

You can define the Replication CDC Service name in either of these ways:

- Update the *svc.conf* file:
  a Navigate to the *etc* directory of the installation folder and use a text editor to open the *svc.conf* file.
  b Update *instance_name* to include the Replication CDC Service Name. For example, if your Replication CDC Service name is ETL_RCS_INS1, enter:

Note All grid engines that start the Replication CDC Service must be in the same subnet.

```
repcdc {
  type = "repcdc";
  container = "inprocess";
  autostart = true;
     config {
              instance_name = "ETL_RCS_INS1";
             }
```

```
}
```

Note The Replication CDC Service name must be unique for each grid engine.

c Save the file as UTF-8 encoded. Otherwise, the grid engine cannot read the file, and may not start.

- Use the repcdcinstancename command line parameter – if you define this parameter value, the grid engine ignores the Replication CDC Service configurations in the *svc.conf* file, and uses the parameter value to start the service. To start a grid engine with the Replication CDC Service name, ETL_RCS_INS1, using the command line, enter:

```
GridNode --repcdcinstancename ETL_RCS_INS1
```

Note The CDC Provider Sybase Replication Server component does not work if Replication CDC Service is not properly configured and running.

Configuring Oracle as a replication source

This section describes the tasks involved in configuring Oracle as a replication source.

Note If you have already configured the Oracle database in your replication environment, you must still perform the tasks specified in section "Configuring the Oracle instance," "Configuring Replication Agent," and "Adding primary database to the replication system," to use ETL.

Installing Replication Agent (RAX) and creating a Replication Agent for Oracle (RAO) instance

1 Install Replication Agent.

2 Copy the downloaded license file to *SYSAM-2_0\licenses* in the installation folder.

3 Start a RAO instance. For example:

```
ra_admin -c rao_inst1 -p 1333 -t oracle
```

4 Include the Oracle JDBC driver jar files to CLASSPATH. For example:

```
set CLASSPATH=%CLASSPATH%;C:\oracle\product
\10.2.0\db_2\jdbc\lib\ojdbc14.jar
```

Configuring the Oracle instance

Connect to the Oracle instance as a system administrator, using SQLPLUS, and perform these configuration tasks:

1 Prepare Oracle to use redo logs

To verify the archive log mode, enter:

```
select log_mode from v$database;
```

If the archive log mode is on, this message displays:

```
LOG_MODE
-------
ARCHIVELOG
```

If the archive log mode is off, this message displays:

```
shutdown immediate;

exit
```

Run the .sqlpus/nolog command to set the archive log mode to on and then enter:

```
connect sys/password as sysdba;
startup mount;
alter database archivelog;
alter database open;
alter system set recyclebin=off;
```

2 Enable supplemental logging for the source table.

```
ALTER TABLE T1 ADD SUPPLEMENTAL LOG DATA (ALL)
COLUMNS;
```

Note It is a *must* to enable supplement logging for the source table in ETL.

3 Add primary key information to the Oracle redo log.

```
alter database add supplemental log data (primary
key,
unique index) columns;
select SUPPLEMENTAL_LOG_DATA_MIN,
SUPPLEMENTAL_LOG_DATA_PK,
SUPPLEMENTAL_LOG_DATA_UI from v$database;
```

If the primary key information is successfully added, this message displays:

```
SUPPLEME SUP SUP
```

```
-------- --- ---
YES      YES YES
```

4 Create Oracle users for Replication Agent and Replication Server, and then grant connect,resource,dba to them;

Note Do not use the Oracle user for Replication Server to perform any DML transactions, as Replication Server does not capture data changes made by this user.

5 Configure Replication Agent. See Configuring Replication Agent.

Configuring Replication Agent

To configure Replication Agent, start the RAO instance and connect to it using isql. Then, perform these configuration tasks:

1 Set the archive log file path of the source Oracle database. Enter:

```
ra_config pdb_include_archives, true
go
ra_config pdb_archive_path, <path-to-oracle-
archive-directory>
go
```

2 Configure connection of Replication Agent to the primary database. Enter:

```
ra_config pds_host_name, <the host name of the
source oracle>
go
ra_config pds_port_number <the port number of the
source oracle>
go
ra_config pds_database_name,<the source oracle
database name>
go
ra_config  pds_username, <the oracle user for
Replication Agent>
go
ra_config pds_password, <password>
go
test_connection PDS
go
```

If the connection is established successfully, this message displays:

```
Type Connection
---- ----------
```

```
PDS  succeeded
```

3 Configure the Replication Agent connection to Replication Server. Enter:

```
ra_config rs_host_name, <the host name of the
Replication Server>
go
ra_config rs_port_number, <the port number of the
Replication Server>
go
ra_config rs_username, <the Replication Server user
for Replication Agent>
go
ra_config rs_password, <password>
go
ra_config rs_source_ds <the current RAO instance
name>
go
ra_config rs_source_db, <the source oracle database
name>
go
```

Note It is a *must* to specify the RAO instance name for ETL.

4 Configuring the Replication Agent connection to ERSSD. Enter:

```
ra_config rssd_host_name <the host name of the
ERSSD>
go
ra_config rssd_port_number, <the port number of the
ERSSD>
go
ra_config rssd_username, <the ERSSD user for
Replication Agent>
go
ra_config rssd_password, <password>
go
ra_config rssd_database_name, <the database name of
the ERSSD>
go
test_connection RS
go
```

If the connection is established successfully, this message displays:

```
Type Connection
---- ----------
RS succeeded
```

5 If the charset of Replication Server is not the same as Replication Agent, update the charset. Enter:

```
ra_config rs_charset, <the charset of the
Replication Server>
```

6 To handle update or delete transactions correctly, configure ltl_send_only_primary_keys as false. Enter:

```
ra_config ltl_send_only_primary_keys, false
```

Note ETL is unable to handle delete or update transactions if you do not perform this step.

7 Initialize Replication Agent. Enter:

```
pdb_xlog init
```

Adding primary database to the replication system

Using isql, connect to Replication Server, and enter:

```
create connection to  < rs_source_ds >.< rs_source_db >
set error class rs_sqlserver_error_class
set function string class rs_sqlserver_function_class
set username <the oracle user for Replication Server>
set password <password>
with log transfer on, dsi_suspended
```

where:

- *rs_source_ds* – is identical to value of the parameter, *rs_source_ds*, in Replication Agent.
- *rs_source_db* – is identical to value of the parameter, *rs_source_db*, in Replication Agent.

Resuming Replication Agent

Replication Agent is now ready to replicate transactions. Start the replication by executing the resume command in Replication Agent:

```
resume
go
```

Verify if Replication Agent is replicating

```
ra_status
go
```

If Replication Agent is setup correctly, this message displays:

```
State        Action
------------ ------------------------
 REPLICATING  Ready to replicate data.
```

Configuring a CDC Provider Replication Server component

1 Drag the CDC Replication Server Provider component onto the Design window.

2 Specify the name of the Replication Server you want to use to capture data changes, along with its user name, and password.

3 Set the replication definition options in the Source Table Options field. See “Setting replication definition options” on page 125.

4 To filter qualified data changes, select one of these capture modes:

- Full – receives all changes and sends them one by one. The output changes are same as the input changes from Replication Server.
- Last – sends only the last received data changes of each row. All changes with the same key values are merged into a single change.

  For example, if there are two changes:

```
1.update test_table set col_1='x' where key_col='A'
2.update test_table set col_1='y' where key_col='A'
```

  If you select Last as the capture mode, the output is:

```
update test_table set col_1='y' where key_col='A'
```

Sybase recommends to set this property to Last, which is appropriate for most incremental loading. If set to Last, the Insert and Update changes are regarded as Upsert changes.

Note If you select Full, and if there are several changes on the same row, you must ensure the sequence while loading to the target database.

5 If capture mode is set to Last, enter the Stage Mode property to specify whether you want the component to stage all received data changes in memory, or in IQ.

Note For a 32-bit grid engine, it is recommended to set the stage mode to IQ.

6 Specify the staged IQ configurations in the "IQ for Stage Mode Options" field, if the Capture Mode is set to Last, and the Stage Mode is set to IQ.

7 In the Ports Options field, specify the OUT-ports details. See "Configuring the OUT-ports" on page 126.

8 Specify any other optional properties. See "CDC Provider Replication Server properties list" on page 127.

9 Create replication. See "Creating and dropping replication" on page 126.

Setting replication definition options

1 To open the CDC Configuration window, click the Source Table Options icon.

2 Enter connection details for the source database.

3 Click Options to select the source table to replicate.

4 Select the Replicate option for each column to replicate.

5 Select the Key option against the column you want to mark as the primary one.

You must select one or multiple columns as the key column to prevent errors during replication creation.

6 Click Save.

Creating and dropping replication

❖ **Creating replication**

Creating replication includes creating the replication definition, replication connections, function strings, and replication subscriptions with "No materialization" as the materialization method. It enables Replication Server to start capturing the source table data changes.

1 Right-click the component and select Create Replication.

2 Confirm that source and target tables are synchronized and click Yes.

On successful creation, the Replication property in the Property window changes to "Created," and the status is written to the repository.

❖ **Dropping replication**

Dropping replication includes dropping the replication definition, replication connections, function strings, and replication subscriptions. It enables Replication Server to stop capturing the source table data changes, and clears all unprocessed data changes of the source table.

1 Right-click the component and select Drop Replication.

2 Confirm that you are dropping the replication. Click Yes.

The Replication property in the Property window changes to "Dropped," and the status is written to the repository.

Configuring the OUT-ports

1 Click the Ports Options icon to open the CDC Provider Ports dialog.

2 To add a port, click the Add Port icon, and enter a name for the new port. Click OK.

Note You cannot change a port name in the CDC Provider Ports dialog. To change the port name, go to the Design window, right-click the port, and select Description.

3 To change the output data from an OUT-port, select an option from the Function drop-down list, and click OK.

4 To remove a port, select it and click the Remove Port icon. There must be at least one OUT-port on the CDC Provider Replication Server component.

CDC Provider Replication Server properties list

The following tables list the required and optional properties of the component.

Required properties

Property	Description
RepServer Name	Specify the Replication Server to use to capture data changes. After Replication is created, this property cannot be modified. If you want to modify it, right-click the component and select Drop Replication.
Rep Database	Object name generated during replication creation. This property is read-only.

Property	Description
Source Table Options	Set the replication definition options: • Interface – identify the method or driver you want to use to connect to the primary data source. • Host Name – identify the primary data source. The options in the Host Name list depend on the selected interface. • User and Password – identify an authorized database user, and protect the database against unauthorized access.If the source database is ASE, the database user requires “sa” or “dbo” permission or “replication_role.” • Database – identify the database to use as data source.If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password. • Schema – identify the schema or owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema. • Table – identify the source table. • Replicate – identify the column to be replicated. **Note** Since ETL does not support Large Object (LOB) data processing, a column with LOB type is not replicated. • Key – identify the columns with keys. Replication Server and ETL use the key to identify each row of the source table. **Note** Replication Server does not allow using column with LOB type as key. The key values should be unique for each row and the key columns must be replicated. • Size – identify the column size. You can change it to an appropriate value. If the source database is Oracle, enter Replication Agent details: • RepAgent Host – Replication Agent name, which must be identical to the configuration value of ‘**rs_source_ds**’ in Replication Agent. • RepAgent Database – Replication Agent database name, which must be identical to the configuration value of ‘**rs_source_db**’ in Replication Agent. • RepAgent Username and Password – user name and password to connect to Replication Agent. After Replication is created, this property cannot be modified. If you want to modify it, right-click the component and select Drop Replication.

Property	Description

Port Options	Specify port type of each OUT-port:
	• Port Name – specify the OUT-port name.
	• Function – specify the data changes to be sent from this port:
	  • Insert – sends insert changes.
	  • Delete – sends delete changes.
	  • Update – sends update changes.
	  • Upsert – sends insert and update changes.
	  • All – sends all data changes.

Capture Mode	Specify whether to send all received data changes or only the last received data changes on each row, to the next component. By default, this property is set to "Last," which is appropriate for most incremental loading.
Replication	Indicates whether replication is created or dropped. The value for this property is generated by ETL and is read-only.

Optional properties

Property	Description
Rep Server User and Password	Specify an authorized Replication Server user name and password.
Stage Mode	Specify whether data changes should be staged in the memory or in an IQ temporary table. If the capture mode property is set to Last, provide a value for the stage mode property. **Note** For a 32-bit grid engine, it is recommended to set stage mode to IQ.
IQ for Stage Mode Options	Specify configuration for staging IQ. This property is enabled only when capture mode is Last, and stage mode is IQ.
Timeout seconds for no data	Specify the wait time, in seconds, for the component before it stops waiting to receive data changes from Replication Server.
Read Block Size	Determine the number of records retrieved by the component in a single step.

Property	Description
Auto Initial Load	Specify whether you want the initial load to take place automatically. Before transferring source table data changes, ETL must perform an initial load to synchronize the source and destination table data, and create replication objects to make Replication Server start capturing data changes. **Note** Since Replication Server does not support materialization for Oracle, this property does not work if the source database is Oracle. While executing or simulating a project: • If this property is selected and Replication is not created, then ETL creates the replication definition, connection, and function strings, and uses the Atomic materialization method to create subscription. This materialization method locks the source table until all source table records are captured by Replication Server as insert operations in a transaction. After this, simulation or execution performs the initial load. • If this property is not selected and Replication is not created, you see an error message. • If this property is selected and execution or simulation fails, right-click the component and select Drop Replication. Try to execute or simulate the project again. If you right-click the component and select Create Replication, while this property is selected, initial load does not take place. **Note** While simulating a project, if this option is selected and you click Reset and Start to restart simulation, CDC Provider Sybase Replication Server component does not receive the source table initial data a second time. Replication Server pushes initial data to ETL only once.
Output Old Value	Select to output old column values of each data change. After Replication is created, this property cannot be modified. If you want to modify it, right-click the component and select Drop Replication.

Property	Description
Transactional	All work performed by the CDC Provider Sybase Replication Server component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.
Save Interval	Specify the amount of time, in minutes, for which Replication Server should save the data changes after it is received by ETL. By default, Replication Server discards each data change after ETL receives it. Set this property to have Replication Server save data changes for a while after ETL receives them. It is necessary to save data changes, since a project or job failure can cause the destination table changes to roll back, if project level transactionality is enabled. In this case, all source table data changes received are not applied to destination tables. In subsequent executions, the data changes that are not processed successfully in the previous execution, that still exist in the Replication Server, are sent to the CDC Replication Server Provider component, along with any new changes. This ensures that the destination tables are updated correctly. If the save interval is too short, some data changes from running a failed project are not sent to the destination components, causing the destination tables to be out of sync with the source table data. If the save interval is too long, duplicate data changes are sent to the component, leading to decreased performance, and increased resource usage.

Transformation components

Transformation components have both IN-ports and OUT-ports, and apply specific transformations to the data in the transformation stream.

Component	Description
Character Mapper	Replaces characters and strings in an input record. Character Mapper applies replacement mapping to all selected attributes.
Copy Splitter	Unconditionally makes a copy of the input data to each of the output ports.
Data Calculator JavaScript	Performs transformations to every record passed to this component between the IN-port and OUT-port.
Data Splitter JavaScript	Splits an incoming data stream based on input values.
SQL Executor	Use this component to execute one of more custom SQL statements against a database server. SQL Executor is a standalone component without IN or OUT ports.

Character Mapper

The Character Mapper is a Transformation component that replaces characters and strings in an input record. Character Mapper applies replacement mapping to all but selected attributes.

Use Character Mapper to replace characters or strings, for example, to change a German umlaut (ä) to ae or Unicode characters.

Configuring a Character Mapper component

1 Drag the Character Mapper component onto the Design window.

2 Link the IN-port of the Character Mapper to the OUT-port of the component that provides the inbound data. Link the OUT-port of the Character Mapper to the IN-port of the component to which you want to direct the outbound data.

You must configure the IN-port structure of the component where you want to direct the outbound data.

3 Open the Character Mapper component window. If necessary, click the "Step to next incoming data buffer" icon on the toolbar to populate the Input and Output content.

4 Add a mapping definition. See "Creating new mapping definitions" on page 133.

Working with the Character Mapper component window

Use the Character Mapper component window to define mapping rules for data that passes between the IN-port and OUT-port.

The Character Mapper component window includes the:

- Current Input Record pane – displays the columns, rows, and content for the record currently at the IN-port.
- Current Output Record pane – displays the columns, rows, and content for the current record as it appears to the OUT-port.
- Mapping definition pane – includes a From column and a To column.

❖ Creating new mapping definitions

1 Click the Insert Mapping icon on the toolbar, or right-click anywhere on the Mapping Definition pane, and select Insert Mapping. A new row is inserted for the new mapping definition.

2 Enter the character combination you want to replace in the From column and the value with which you want to replace it in the To column. See Mapping notations.

Character Mapper applies the rule, and displays the results in the Current Output Record pane

- To edit the record currently displayed in the Current Input Record pane, click a row in the Current Input Port Content pane and make changes. The values in the Current Output Record pane and the selected row in the Current Outport Port pane are also updated.
- To delete a mapping definition, right-click the mapping definition and select Remove Mapping.
- To change the order of the mapping definitions, select the Move row up and Move row down icons on the toolbar.

3 Character Mapper applies the mapping, and updates the Output results as soon as you write the rule. Click the Enable auto refresh of output values icon on the toolbar to toggle this kind of automatic synchronization.

Note Character Mapper applies mappings to all columns and all rows. To exclude columns from a character mapping, click Exclude in the Properties window, and select the columns you want to exclude.

Mapping notations

The From and To values can contain any combination of characters in these notations:

Type	Syntax	Example (@)
Character	%	@
ASCII decimal	<%>	<64>
ASCII hexadecimal	<0x%>	<0x40>
Unicode decimal	<u%>	<u0064>
Unicode hexadecimal	<u0x%>	<u0x0040>

Note % represents the respective character code.

These notations also allow special characters to be mapped. Examples for each supported notation include:

Mapping	From	To
Umlaut internationalization	Ä	Ae
Keyword translation	kunde	customer
Delete CR LF from string	<13><10> or <0x0D><0x0A>	
Replace the lira currency symbol with the euro currency symbol	<u8356> or <u0x20A4>	<u8364> or <u0x20AC>

Reusing mapping definitions

You can save character mapping definitions to a file, allowing then to be reused in other projects. See Exporting mapping definitions and Importing mapping definitions.

❖ **Exporting mapping definitions**

1 On the Character Mapper component window, click the Export character mapping icon on the toolbar.

2 Provide a file name and click Save. Character Mapper saves the file without an extension.

❖ **Importing mapping definitions**

1. In the Character Mapper component window, click the Import character mapping icon on the toolbar.
2. Select the file you want to import and click Open.

Character Mapper demos

Sybase ETL includes a demonstration for the Character Mapper component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Transform | Character Mapper.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select Demo Character Mapper.

Copy Splitter

The Copy Splitter component unconditionally copies input data to each output port. Unlike the Data Splitter, the Copy Splitter does not include any port expressions.

Using the Copy Splitter eliminates the need to invoke JavaScript, and reduces the cost of condition evaluation, thus improving performance.

By default, the Copy Splitter component has two output ports.

Configuring a Copy Splitter component

1. Drag the Copy Splitter component onto the Design window.
2. Link the Copy Splitter IN-port to the OUT-port of the component that provides the inbound data.
3. If required, create additional OUT-ports. See “Adding new OUT-ports.”
4. Link the Copy Splitter OUT-ports to the IN-ports of the components to which you want to direct outbound data.

Managing Copy Splitter OUT-ports

❖ **Adding new OUT-ports**

Although the component is configured with two OUT-ports, you can create additional OUT-ports. New ports are identified by the name.

1 Right-click the component and select Add Output Port.

2 Enter a name for the new OUT-port. Click OK.

❖ **Removing OUT-ports**

1 Right-click the port to remove and select Remove Port.

If the OUT-port is linked to another component, Remove Port is disabled. You must delete the link before deleting the port.

Note You cannot delete all ports. The component must have at least two OUT-ports.

❖ **Managing port structures**

- See “Managing port structures” on page 90 for information on how to manage port structures and port attributes.

Data Calculator JavaScript

Data Calculator JavaScript is a Transformation component that lets you define rules that apply transformations to records that pass between IN-ports and OUT-ports. You can, for example, use Data Calculator JavaScript to define rules that transform port attributes or add rules that create new attributes.

Data Calculator can also perform lookups at the attribute level. You provide lookup data at special lookup ports.

Without lookups, Data Calculator JavaScript does not impact the simulation sequence. With lookups, all data is read into the lookup ports before the data at the main port is processed.

Configuring a Data Calculator JavaScript component

1 Drag the Data Calculator JavaScript component onto the Design window.

2 If necessary, link the IN-port of Data Calculator to the OUT-port of the component that provides the inbound data.

3 If the Data Calculator component window is open, click Save to close the window.

4 In the Design window, right-click the OUT-port of Data Calculator, select Assign Structure and select one of these options:

Option	Action
IN	Create an OUT-port structure that matches the IN-port structure of the Data Calculator.
Copy Structure	Open a window that allows you to apply an existing port structure to the OUT-port.

5 In the Design window, double-click Data Calculator JavaScript, or, in the Properties window, click the Rule icon.

6 Select one of these options:

Options	Action
Yes	Create a default mapping by order. Select this option to create a set of transformation rules that maps each attribute in the IN-port to a corresponding attribute in the OUT- port.
No	Define your own IN-port to OUT-port mappings. Select this option to manually map attributes in the IN-port to a corresponding attribute in the OUT-port. See “Mapping port attributes” on page 138.

Working with the component window

The Data Calculator component window provides tabular and graphic views of the data stream. You can also use this window to map port attributes and define transformation rules.

Tabular view

The tabular view shows the current record structure, port attributes, and transformation rules. It includes:

- Current Input Record pane – identifies each attribute and attribute value in the current record at the IN-port. All IN-port attributes include an “IN.” prefix.

- Transformation Rules and Current Output Record pane – includes a column and row for each transformation rule, and columns and rows for each OUT-port attribute. Default transformation rules map each IN-port attribute to a corresponding OUT-port attribute. By default, OUT-port values reflect IN-port values. Changing an attribute value or transformation rule changes OUT-port values.

  If the transformation rule is a functional expression, you must enter it on a single line. The returned value is, by default, assigned to the corresponding OUT-port attribute. However, if you enter a multiline functional expression, the text is interpreted as a script, and you must set the OUT-port attribute value.

- Input Port Content pane – shows the current set of records available at the IN-port.
- Output Port Content pane – shows the current set of records available at the OUT-port.

Graph view

The graph view shows the current mapping between the IN-port and OUT-port attributes.

- Input Port Content pane – shows the current set of records available at the IN-port.
- Output Port Content pane – shows the current set of records available at the OUT-port.

Note After you apply a transformation rule to an IN.attribute, the mapping line between the IN.attribute and the OUT.attribute no longer appears.

❖ **Mapping port attributes**

Although the Data Calculator component creates column-to-column mapping between the IN-port and OUT-port as a default option, there may be times when you want to map port attributes individually. To create your own mappings, use the graphic view.

1 On the Data Calculator component window, click the Graph tab.

2 Map the IN-port and OUT-port in one of these ways:

- Select Mapping in the menu bar, and select one of these predefined mapping sequences:

- Create mapping by Order – sequentially maps the port attributes of the IN-port and OUT-port.

  Note If the number of attributes is different, some port attributes are not mapped.

- Create mapping by Name – maps the port attributes of the IN-port and OUT- port according to their names.
- Create mapping by Name Case Sensitive – maps the port attributes of the IN-port and OUT-port according to their case-sensitive names.
- Create mapping by prefix – maps the port attributes of the IN-port and OUT-port by name, ignoring the specified prefixes.
- Create mapping by Best Match – maps the port attributes of the IN-port and OUT-port that sound alike.

- Connect the IN-port and OUT-port attributes individually.

The component is now ready to be used and can forward records from the IN-port to the OUT-port.

Displaying transformation results

The Data Calculator immediately displays the result of transformation rules, allowing you to verify incoming data, test transformation rules, and view the effect of the rules on data output.

By default, OUT-port values reflect IN-port values. Manually changing an IN-port attribute value affects only the data of the IN-Port or OUT-port buffer, allowing you to test transformation rules, and to see the results in the current output record. This is a convenient way to create test cases for transformation rules.

Use the Transformation Rule column to add, modify, or delete transformation rules. You can also edit single-line functions by changing the current attribute input field.

See “Using the JavaScript Editor and Debugger” on page 67 for information about creating complex procedural transformations.

Note The graphic view mirrors the actual port structures. You cannot add or delete attributes in the graphic view.

Managing transformation rules

- To add a transformation rule, right-click anywhere in the Transformation Rule or Current Output Port column, and select Insert. You can now use the added rules for further assignments or calculations.
- To delete a transformation rule, right-click a row in the Transformation Rule column, and select Remove.
- To change the order of the transformation rules, right-click the row in the Transformation Rule column, and select Up or Down.
- To add missing output attributes, click Mapping, and select Add missing output attributes.

Transformation rules are processed in sequential order. The processing starts with the first transformation rule of the list.

Simulating Data Calculator

Data Calculator is designed so you can see the changes applied to data as it moves through the transformation rules. You may find this useful, for example, to see how changing a transformation rule affects outgoing data. Depending on the status of the Auto-Synchronization button, a transformation rule is immediately applied to the entire set of IN-port records after the rule is entered.

- Toggling auto synchronization – auto synchronization immediately applies all changes to the transformation rules to the current set of records at the IN-port.

  Note If Auto-Synchronization is disabled, you can manually trigger the processing of the IN-port data by selecting the Step option.

- Manually apply all transformation rules to all current records at the IN-port.– by clicking the Step icon on the toolbar.
- Fetching another set of records – by clicking the Step through the next incoming data buffer icon on the toolbar.
- To step through IN-port records –
  - Click the appropriate record control on the toolbar.
  - Click Navigate and select the appropriate option.
  - Select a record from the Input Port Content list.

- Search for keywords in the transformation rules – by clicking the Search the Content of Transformation Rules icon on the toolbar

Note The values shown on the Current Input Record area are updated as you change the current record.

- Highlight null and empty values – by clicking the Highlight NULL-Values and Empty Values icon on the toolbar.

Using lookups in Data Calculator

Data Calculator performs lookups at the attribute level. You must enter lookup data at special lookup ports.

Adding Lookup Ports

To add a lookup port to Data Calculator, connect the OUT-port of the data providing component directly to the Data Calculator component (not a port). A lookup port is automatically created and connected. Alternatively, right-click the Data Calculator and select Add Input Port. Connect the OUT-port of the data providing component with the new port. You can add an unlimited number of lookup ports.

Preparing the lookup data

Each lookup port must have at least two attributes. The first attribute represents the key. All other attributes represent return values.

To look up compound keys, concatenate the key values within a preceding component and use the same kind of concatenation on the key expression for the lookup.

Setting general lookup options

Use the Lookup Options property to configure the lookup. The Properties window displays a list of all lookup ports and current option values.

- Lookup Name – is inherited from the associated port and cannot be changed here. To change the name of a port, select Description from the port menu.
- Lookup Size – to optimize memory allocation and lookup performance, enter the estimated number of lookup records. See “DB Lookup” on page 150.

- Lookup Empty / Null – empty and null are normally handled as "unknown" keys, thus returning the lookup default value. If empty or null values are valid keys for your lookup, you can enforce looking up the values for these keys by activating this option.

Building lookup rules

To set up lookup rules, click the Tabular tab in the Data Calculator window. If lookup ports are available, an additional Lookup column appears.

For each lookup rule, provide:

- Key Expression – is the value to search for in the first column of the lookup list. Enter the key expression (for example IN.PR_ID) as a transformation rule.
- Return Value – since lookup lists can have more than one return value column, you must specify which value to return, if the key is found. Select the associated port attribute from the pop-up menu on the Lookup column. For example: LOOKUP1>>LOOKUP1.PR_NAME.

  Note Although return values are selected and displayed by name, the lookup internally uses the column number. This means you must review your lookup rules whenever the lookup port structure is modified, especially after attributes are added or removed.

- Output Variable – the lookup return value is assigned to this variable. You can select a variable from the Output Port column (for example, OUT>>OUT.PR_NAME).
- (Optional) Default Expression – a lookup returns null if the given key value is not found. To return a different default value, enter an expression in the Lookup Options window. To open the Lookup Option window, click the icon in the Lookup column in the transformation Rules and Current Output Record panel.

Data Calculator JavaScript demos

Sybase ETL includes several demonstrations for the Data Calculator component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Transform | Data Calculator - JavaScript.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects.

For demos without lookup, select:

- Demo Transfer U.S. Products
- Demo Transfer German Products
- Demo Data Calculator

For demos with lookup(s), select:

- Demo Transfer German Customers
- Demo Transfer German Sales
- Demo Transfer U.S. Customers

Data Splitter JavaScript

The Data Splitter JavaScript component allows you to easily filter and distribute input data.

Configuring a Data Splitter JavaScript component

1 Drag the Data Splitter JavaScript component onto the Design window.

2 Link the Data Splitter IN-port to the OUT-port of the component that provides the inbound data.

3 Link the Data Splitter OUT-ports to the IN-ports of the components to which you want to direct outbound data.

4 Select Mutually Exclusive Rules if you want port conditions to behave as if they are mutually exclusive. If this option is selected, ports are evaluated based on a user-specified order, instead of the order in which they appear on the component. When conditions are defined as mutually exclusive, the splitter writes a given record to 0 or 1 OUT-port. The first OUT-port that has a matching condition receives the input record. No subsequent port conditions are evaluated. If a condition does not match, the record is not written to any OUT-port.

Note The mutually exclusive port conditions are always evaluated in the order specified by the sequence number associated with each port condition.

5 Double-click the Data Splitter JavaScript component.

6 Add the conditions you want to use to direct the data flow:

   a Double click the port you want to add the conditions to. Alternatively, click the Edit condition icon on the port, or right-click the port and select Edit condition.

   b On the Condition window, define the conditions for each column that you want to apply.

   c Click Save.

Splitting inbound data

Adding the Data Splitter component to a project opens a component window that displays inbound data attributes and OUT-port conditions.

The Data Splitter component is configured with two OUT-ports.
Both port conditions are preset to 1. Since this condition is always true, regardless of the current values of the IN-port, the Data Splitter copies all incoming records to both OUT-ports.

The IN-port data buffer is initially empty, and only the inbound data attributes are visible. The OUT-port structures match the IN-port structure.

To populate the input attributes, click the “Step to the next input buffer” icon on the toolbar. Input data appears in the upper part of the component window.

Since the OUT-ports share the same port structure as the IN-port, selecting any record in the upper window causes the OUT-port to indicate whether the record meets the port condition. When a record meets port conditions, the OUT-port color is green. When a record does not meet port condition, the OUT-port color changes to red.

Inclusive port conditions

If the splitter mode is inclusive that is you have not defined mutually exclusive port conditions, each input record is tested against each port condition. For every matching port condition, a copy of the current record is written to the OUT-port.

Exclusive port conditions

If you have defined mutually exclusive port conditions, ports are evaluated based on a user specified order, instead of the order in which they appear on the component. The conditions are evaluated in the order of first to last. The first output port with a matching condition receives the input record. No further port conditions are evaluated. You can change the evaluation order of conditions by clicking the “Edit evaluation order” icon and then swapping rows in the Evaluation order window. To reset the evaluation order to default, click the “Reset evaluation order” icon.

Note The number of records that Data Splitter forwards to the OUT-ports can differ from the number of incoming records. If the port condition is not defined as mutually exclusive, when a single record matches more than one port condition, it is available on all of these ports. Records that do not match any of the conditions are removed from the data stream. For mutually exclusive port conditions, a single input record can match only one OUT-port. Records that do not match are not written to any OUT-port.

Customizing port conditions

You can assign a condition to each port. A condition consists of one or more expressions. Multiple expressions are concatenated by operators. When a condition is evaluated, the result is either true (1) or false (0).

❖ **Modifying port conditions**

1 Double-click the component. In the component window, right-click the port and select Edit Condition.

2 Create the conditions you want to apply to the port. You can:

- Manually enter the conditions in the text area of the Condition window.

- Drag and drop the variables and functions you want to add to your condition from the left pane to the text area. The Variables tab lists all the variables that you can use, and the Functions tab lists all available functions and operators that you can add to the condition.
- Right-click the text area, and select the variables you want to add to the condition.

❖ **Adding new OUT-ports**

Although the component is configured with two OUT-ports, you can create additional OUT-ports. New ports are identified by the name.

1 Click the Add new port icon on the port toolbar.

2 Enter a name for the new OUT-port. Click OK.

❖ **Removing OUT-ports**

1 Select the port you want to remove.

2 On the port toolbar, click the Remove selected port icon. Alternatively, right-click the port and select Delete.

Special port conditions

Port conditions determine how Data Splitter distributes records.
In non-exclusive mode you can have multiple ports having conditions:

- 1 – true. All records are forwarded to this port, including records that match any other port condition.
- (empty) – All records that do not match any other condition are forwarded to this port.

In exclusive mode you can have a single port having the condition:

- 1 – true. All records that do not match any preceding port condition are forwarded to this port.

Note This condition needs to be the last in the evaluation order. Ports with empty conditions are invalid.

Data Splitter JavaScript demos

Sybase ETL includes several demonstrations for the Data Splitter component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Transform | Data Splitter - JavaScript.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select:

- Demo Data Splitter
- Demo Text Data Sink Delimited/Fixed

Note Demos do not display the mutually exclusive port condition functionality.

SQL Executor

The SQL Executor component allows you to execute one or more custom SQL statements against a database server. SQL Executor is a standalone component without IN-ports or OUT-ports. You can place SQL Executor in a project separate from other components, or in a single project containing one or more SQL Executor components. For example, you can use SQL Executor to:

- Load data from a source table to a text file (with a format that IQ can support), if the source table allows extracting data into a file by using a SQL statement.
- Load data from a text file into the target IQ database, in a single transaction, using the Load Table command.

Configuring a SQL Executor component

1 Drag the SQL Executor component onto the Design window.

2 In the Properties window, specify the properties for the component. See SQL Executor properties list, below.

SQL Executor properties list

The following tables list the required and optional properties of the SQL Executor component.

Required properties

Property	Description
Interface	Identify the method or driver to use to connect to the data source.
Host Name	Identify the data source. The options that appear in the Host Name list depend on the interface you select.
Execute Script	Specify the custom SQL script to be executed.

Optional properties

Property	Description
User and Password	Identify an authorized database user and protect the database against unauthorized access.
Execute Success Script	Specify the custom SQL to be executed, if the Execute Script runs successfully.
Execute Error Script	Specify the custom SQL to be executed, if the Execute Script fails.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.

Property	Description
Schema	Identify the schema or owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in the schema you specify.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Transactional	All work performed by the SQL Executor component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

Lookup components

A lookup operation looks up a value that corresponds to a key in a lookup table that contains a list of key and value pairs. A static lookup table can be cached during the execution of a project, or a lookup can be performed uncached and dynamically.

Component	Description
DB Lookup	Looks up values in a database. The lookup data is specified by the result set of a query that returns exactly two columns: the lookup key and the lookup value. You can assign the value returned (lookup value) by the lookup to any attribute of the current record. The lookup table is cached during project execution. Changes applied to the underlying database during project execution have no effect on the lookup result.

Component	Description
DB Lookup Dynamic	Performs a dynamic lookup by referencing the key value in the query `WHERE` clause. DB Lookup Dynamic does not cache lookup information, and performs one SQL lookup for each record that passes the component. During project execution, the lookup table data may be modified by concurrent database users (or even within the same project). In this case, a database lookup that is not dynamic may search for invalid data. Use DB Lookup Dynamic to ensure you locate the current value.

DB Lookup

The DB Lookup component looks up values in a database. The lookup data is specified by the result set of a query that returns exactly two columns—the lookup key and the lookup value.

You can assign the return value (lookup value) by the lookup to any attribute of the current record. DB Lookup caches the lookup table during project execution. Changes that are applied to the underlying database during project execution have no effect on the lookup result.

Configuring a DB Lookup component

1 Drag the DB Lookup component onto the Design window.

2 In the Design window, connect the DB Lookup's IN-port with the OUT-port of the component that provides the inbound data.

3 In the Properties window, specify a valid interface and host name.

See "DB Lookup properties list" on page 151 for specific field requirements.

4 Specify the Key Attribute holding the value to look for and Value Attribute to take the lookup result value. You can replace a value by choosing the same attribute for both.

5 In the Properties window, click the Query icon to open the Query window.

6 On the Query window, create and save the query to retrieve the lookup data. Design your query to return the lookup key and the lookup value from the source table.

Example

Assume that you want to replace the product number used for German products by the product number used in the U.S. The German products are in the table PRODUKTE(PR_NUMMER, PR_NAME, PR_PREIS). The IN-port of the DB Lookup component contains those three attributes.

The table to perform the lookup of the U.S. product number is LOOKUP_PRODUCTS(SOURCE, DESTINATION). The SOURCE column contains the German product numbers and the DESTINATION column contains the U.S. product number.

If no value for the German PR_NUMMER can be found in the LOOKUP_PRODUCTS, the current PR_NUMMER is replaced by the string "INVALID". A successful lookup replaces the German product number by the corresponding U.S. number.

To set up the DB Lookup Component for this example, select:

- Key Attribute – PR_NUMMER
- Value Attribute – PR_NUMMER
- Default Value – INVALID
- Query – select SOURCE, DESTINATION FROM LOOKUP_PRODUCTS

DB Lookup properties list

The DB Lookup properties list identifies the connection parameters and other properties you define on the Database Configuration window.

Required properties

Property	Description
Key Attribute	Select a Key Attribute from the list of IN-port attributes. This attribute corresponds to the first column of the lookup table.
Value Attribute	Select the attribute to receive the value found by the lookup from the value attribute list. The lookup value returned overwrites any existing value. Both Key Attribute and Value Attribute may refer to the same attribute of the record structure, thus allowing a key to be overwritten with its corresponding value.
Interface	Specify the method or driver to use to connect to the data source.

Property	Description
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.
Query	Create a query that retrieves information from the data source. Use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer. See “Query Designer” on page 51.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Default Value	Assign a default value to the value attribute. DB Lookup uses this value when it cannot find the key value in the lookup table.
Use Key Value	Assign the key value to the value attribute instead of the default, if the lookup fails.
Lookup Empty/Null Keys	Perform a lookup for empty or NULL key values. Otherwise, the selected default method applies.
Lookup Size	Specify the estimated number of lookup records, to optimize memory allocation and lookup performance.
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (“.”) as the decimal separator.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Lookup demos

Sybase ETL includes several demonstrations for the DB Lookup component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Lookup | DB Lookup.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects. Then select:

- Demo DB Lookup
- Demo Transfer German Products

DB Lookup Dynamic

The DB Lookup Dynamic component performs a dynamic lookup by referencing the key value in the query WHERE clause. Unlike the DB Lookup component, DB Lookup Dynamic does not cache lookup information, and performs one SQL lookup for each record that passes the component.

During project execution, the lookup table data may be modified by concurrent database users (or even within the same project). In this case, a database lookup that is not dynamic may search for invalid data. Use DB Lookup Dynamic to ensure you locate the current value.

Another typical use for this component is a lookup table that exceeds the memory available on the local machine. The DB Lookup Dynamic component provides a slower-performing lookup, but requires no cache memory, as it performs the lookup on a record-by-record basis.

Configuring a DB Lookup Dynamic component

1 Drag the DB Lookup Dynamic component onto the Design window.

2 In the Design window, connect the DB Lookup's IN-port with the OUT-port of the component that provides the inbound data.

3 In the Properties window, specify the interface and host name.

See "DB Lookup Dynamic properties list" on page 155 for specific field requirements.

4 Specify the Key Attribute holding the value to look for and Value Attribute to take the lookup result value. You can replace a value by choosing the same attribute for both.

5 Specify a Lookup Key Value to be used when designing and testing the lookup Query.

6 Click the Query icon to open the Query window.

7 On the Query window, create and save the query to retrieve the lookup data. Design your query to return the lookup value from the source table. Use the predefined variable Lookup in the where clause of your Query as a placeholder for the Lookup Key. See "Using SBN expressions" on page 66.

❖ **Resetting default lookup key values**

During simulation the Key Attribute values from the incoming data stream are assigned to the predefined variable Lookup. To reassign the specified Lookup Key Value to this variable:

1 Right-click the DB Lookup Dynamic component.

2 Select Reset Lookup Key value.

Example

Assume you want to replace the product number used for German products by the product number used in the U.S. The German products are in the table PRODUKTE(PR_NUMMER, PR_NAME, PR_PREIS). Your IN-port for the DB Lookup Dynamic component therefore contains these three attributes.

The table to look up the U.S. product number is table LOOKUP_PRODUCTS(SOURCE, DESTINATION). The SOURCE column contains the German product numbers and the DESTINATION column contains the U.S. product number.

If no value for the German PR_NUMMER can be found in the LOOKUP_PRODUCTS, the current PR_NUMMER is replaced by the string "INVALID". A successful lookup replaces the German product number with the corresponding U.S. number.

To set up the DB Lookup Dynamic component for this example, select:

- Key Attribute – PR_NUMMER
- Value Attribute – PR_NUMMER
- Default Value – INVALID

- Query – select DESTINATION FROM LOOKUP_PRODUCTS, where SOURCE = '[Lookup]'

DB Lookup Dynamic properties list

The following tables list the required and optional properties of the DB Lookup Dynamic component.

Required properties

Property	Description
Key Attribute	Select a key attribute from the list of IN-port attributes. This attribute populates the placeholder variable Lookup.
Value Attribute	Select the attribute to receive the value found by the lookup from the value attribute list. The lookup value returned overwrites any existing value. Both Key Attribute and Value Attribute might refer to the same attribute of the record structure therefore allowing the overwriting of a key with its corresponding value.
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.
Query	Create a query that retrieves information from the data source. Use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer. See "Query Designer" on page 51.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Default Value	Assign a default value to the value attribute, in case the key value was not found in the lookup table.
Use Key Value	Assign the key value to the value attribute instead of the default, if the lookup fails.
Lookup Empty/Null Keys	Perform the lookup even for empty or NULL key values. If not selected, the default method applies.
Lookup Key Value	Specify a value for the lookup key populating the Lookup variable for testing the query at design time

Property	Description
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example, `2005-12-01 16:40:59.123` Numbers are converted using a period (“.”) as the decimal separator.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Lookup Dynamic demos

Sybase ETL includes a demonstration for the DB Lookup Dynamic component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Lookup | DB Lookup – Dynamic.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects, and then select Demo DB Lookup Dynamic.

Staging components

Staging components have at least one IN-port and one OUT-port, and apply specific transformations to the data in the transformation stream.

Component	Description
DB Staging	Loads incoming data streams into a single staging area. DB Staging buffers all incoming data, then creates an outgoing data stream, which represents the result set of a given select statement.

DB Staging

DB Staging is a Staging component that loads incoming data streams into a single staging area. DB Staging buffers all incoming data, then creates an outgoing data stream, which represents the result set of a given select statement.

You can create staging tables based on the OUT-port structure of the preceding component. Although many transformation components are designed to work on a record-by-record basis, the staging component works in two phases:

- Phase 1 – collect all records from the preceding components.
- Phase 2 – run the query and provide the records of the result set in blocks of a given size.

You can use staging components to perform sorts or aggregations by using ORDER BY or GROUP BY clauses in the Query property. Data from heterogeneous sources can be joined by loading them into multiple tables of the staging database. You can also use the DB Staging component to create an intermediate image of the transformation for further inspection or processing.

Note In simulation, the DB Staging component first retrieves all data from the original data sources, then acts as a new data source for subsequent components. The component allows the Read Block Size value of the original source components to be overwritten.

Configuring a DB Staging component

1 Drag the DB Staging component onto the Design window.

2 Connect the DB Staging IN-port to the component that provides the inbound data.

To add input streams to the DB Staging component, you can drop connections from the data providing component on the staging component. The ports are automatically created by the component.

3 In the Properties window, add the Connection Parameters to the database where you want to add staging tables.

Specify a valid interface and host name to create the connection. See "DB Staging properties list" on page 158 for specific field requirements.

Note If the staging tables you are going to use already exist, skip the next step.

4 In the Design window, right-click the DB Staging component and select one of:

- Create Staging Table from Input – select the appropriate Port structure, and click OK. Enter a name for the new table.
- Create Staging Table from Port – enter a name for the new table and select an appropriate port structure. Click Apply.

5 In the Add table window, verify that the new table information is correct and click Create.

6 In the Properties window, click the Stage Options icon.

The Stage Options window lets you define Truncate Table and the Write Block Size options for each staging table.

7 In the Properties window, click the Query icon and create a query to select data from the staging area.

❖ **Updating port structure with database changes**

- See "Updating port structure with database changes" on page 92.

DB Staging properties list

The following tables list the required and optional properties of the Data Staging component.

Required properties

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.

Property	Description
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.
Stage Options	Set the staging options.
Query	Create a query that retrieves information from the data source. Use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer. See “Query Designer” on page 51.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Read Block Size	Determine the number of records retrieved by the component in a single step.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
PreOutput Processing SQL	Specify an additional SQL script that is executed after all the data from the transformation flow has been loaded into the tables associated with the input ports, and before the query resultset is retrieved. This property enables you to modify or update the staging tables before the output is created.
Post-processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.

Property	Description
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (".") as the decimal separator.
IQ Lock Table in Exclusive Mode	Lock the target table and prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. IQ Lock Table in Exclusive Mode also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait, before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.

Property	Description
Load Stage Path	Specify a data file path. The load stage file must reside on the same machine as the IQ Server. When using a Sybase IQ database, if you specify a Load Stage Path, the component uses the LOAD TABLE statement instead of using SQL statements. This leads to faster performance. **Note** You need not specify the Load Stage Path when client-side loading is enabled for your Sybase IQ 15.0 staging database with ODBC interface. The IQ server automatically uses its default LOAD TABLE statement to add records from files located on the remote host machines into the Sybase IQ table. To create a pipe, specify pipe:// as the Load Stage Path parameter. A pipe is not used if the Load Stage Path is blank. If you use named pipes on UNIX or Linux, the ETL Server and the IQ Server must reside on the same machine. This is not a requirement for Windows.
Load Stage (Server)	Specify the server path to the data file or, leave it empty when using a pipe. If the Sybase IQ server must use a different path to the temporary data file than specified in the Load Stage property, enter it here. **Note** You do not have to specify this property if the file is on the same machine as the grid engine and client-side loading is enabled for your Sybase IQ 15.0 staging database with ODBC interface. If client-side loading is enabled, the IQ server automatically uses its default LOAD TABLE statement to add records from files located on the remote host machines into the Sybase IQ table.

Property	Description
Create Tables	Specify if you want to automatically create tables based on the input port structures at runtime. Select: • None (Default) – if you do not want tables to be created automatically. If a specified table does not exist, an error is thrown. • Non-existing – if you want to create tables that do not exist, based on the structure of the associated input port. • All – if you want all tables associated with input ports to be dropped and recreated based on the port structure. **Note** This property applies to all tables associated with input ports. It cannot be specified at the table level.
Drop Tables	Specify if you want the tables created at runtime, to be removed after the project finishes processing. This property applies to all tables associated with input ports. It cannot be specified at the table level.
Transactional	All work performed by the DB Staging component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See "Job components" on page 210 and "Enabling transactionality for projects and jobs" on page 18 for information on the "Propagate Rollback" property.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See "Database connection settings" on page 94.

DB Staging demos

Sybase ETL includes a demonstration for the DB Staging component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Staging | DB Staging.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects and then select Demo DB Staging.

Destination components

Destination components (also called data sinks) write data to specific targets. This component type has one IN-port and no OUT-port.

Component	Description
DB Bulk Load Sybase IQ	DB Bulk Load Sybase IQ performs bulk operation on a Sybase IQ table. Use this component to manipulate records of a table in a Sybase IQ database based on the records from the IN-port of the component.
DB Data Sink Delete	Removes records from the destination table that match the incoming values of a selected key.
DB Data Sink Insert	Adds records from the IN-port of the component to a database table. You can exclude attributes or assign default values to determine the records you insert into the table.
DB Data Sink Update	Updates or overwrites all records that match a selected key. This component does not insert new records.
Text Data Sink	Writes transformation results to a text file in a delimited or fixed-length format.

Preconditions for using DB Data Sink components for bulk loading

You must enter the Load Stage Path for the DB Data Sink components to support IQ bulk loading. To support IQ bulk loading with client-side loading, the DB Data Sink components must meet these conditions:

- Sybase IQ version must be 15.0 or above.
- The value of the “allow_read_client_file” option must be set to “on”. See “Enabling client-side load support” on page 165.
- Interface must be ODBC.
- Load Stage Path must not be specified.

Additionally:

- The Insert options should not have a SQL Insert value specified in case of the DB Data Sink Insert component.
- The Update options should not contain a SQL UPDATE SET clause, and none of the NON-NULLABLE column should be omitted from the list of update columns, in case of the DB Data Sink Update component.

DB Bulk Load Sybase IQ

DB Bulk Load Sybase IQ is a destination component that performs bulk operation on a Sybase IQ table. Use this component to manipulate records of a table in a Sybase IQ database based on the records from the IN-port of the component.

Configuring a DB Bulk Load Sybase IQ component

1 Drag DB Bulk Load Sybase IQ onto the Design window.

2 Connect the IN-port of DB Bulk Load Sybase IQ to the OUT-port of the component that provides the inbound data.

3 On the Database Configuration window, add the Sybase IQ connection parameters.

4 Click the Destination icon and select the table where you want to load the inbound data. To write to a new destination table, see "Adding new Sybase IQ destination tables" on page 165.

5 Click the Load Stage icon. You can either:

   - Select or enter the file name you want to use as a temporary data file and click Save, or,
   - Add a pipe name in the Load Stage field (syntax: `pipe://`).

     See "DB Bulk Load Sybase IQ properties list" on page 170 for specific field requirements.

6 Click Finish.

To add DB Bulk Load IQ to your project:

- Sybase IQ must be up and running before you add the component to your project. You can increase performance if you run both ETL Server and the IQ database on the same machine; however, this is not required.

- To connect to the target database on IQ, select a valid host name and interface. See “DB Bulk Load Sybase IQ properties list” on page 170 for specific field requirements.
- To load the data into a new IQ table, you can create a destination table based on DB Bulk Load’s IN-port or the structure of any available port in the project. See “Adding new Sybase IQ destination tables” on page 165.
- To customize the script, right-click the DB Bulk Load IQ component, and select Generate Load Script. Click the Load Script icon in the Properties window, edit, and save your script.

❖ **Updating port structure with database changes**

- See “Updating port structure with database changes” on page 92.

Adding new Sybase IQ destination tables

You can write the inbound data to an existing table, or add a new destination table based on existing ports in the project.

❖ **Adding a destination table based on IN-port**

1 Right-click the component and select Add Destination Table from Input.

2 Enter a name for the new table.

3 Click OK.

❖ **Adding a destination table based on an existing port**

1 Right-click the component and select Add Destination Table from Port.

2 Enter a name for the new table and click OK.

3 Select the port you want to use to create the table. Click Apply.

Enabling client-side load support

You can use this component to add data into the Sybase IQ table, from files that are located on a different host machine than Sybase IQ. You need not install Sybase ETL and Sybase IQ on the same machine; ETL Server and Sybase IQ can communicate in a networked environment, allowing you to bulk-load from a remote machine in a single step. To support client-side:

- Install the Sybase IQ 15 client on the same machine as ETL Server.
- Install the Sybase SQL Anywhere 11 ODBC driver on the same machine as ETL Development and ETL Server.

- The target IQ database version must be Sybase IQ 15.0.
- On each Sybase IQ 15.0 server, enable the allow_read_client_file and allow_write_client_file options. To set these options:

  a From Sybase Central™, connect to the Sybase IQ 15.0 server.

  b Right-click the Sybase IQ server database name and select Options.

  c Select allow_read_client_file and allow_write_client_file options and change their values to On. By default, the value is Off.

  d Enable the allow_read_client_file server option property using the isql or dbisql utility:

```
set option allow_read_client_file=on
GRANT READCLIENTFILE TO <group | user>
```

Once you have completed these prerequisites, select Use IQ Client Side Load in the Properties window of the component. Also, select ODBC as the interface, or you may encounter errors loading data from remote host machines. Client-side loading works only with ODBC when the ODBC driver being used is an IQ 15 ODBC driver.

Note If you select Use IQ Client Side Load to enable bulk loading of data into the IQ database from files located on client machines, provide a file path name instead of a pipe name in the Load Stage property field. Client side loading is not supported using Load Stage pipe names.

Configuring multiple writers for loading data

Sybase ETL supports the multiple writer functionality that is available in Sybase IQ 15.0. This functionality allows you to add multiple writers for loading data into an IQ database. Multiple writers allow parallel loading of Sybase IQ tables and is faster than sequential loading. You can use multiple writers if you have:

- Selected more than one table from the source database and you are migrating to more than one table in the target IQ database.
- Created a job with a multiproject component involving multiple tables, or if you have selected multiple projects that are linked to parallel execution topology for job execution.

To use the multiple writer functionality, you must have these permissions in the target IQ database:

Object name	Type	Required permission
ETL_MULTIPLEX_STATE	Table	create
ETL_MULTIPLEX_VERSION	Table	create
sp_iqstatistic	Stored Procedure	execute

Note ETL creates two tables ETL_MULTIPLEX_STATE and ETL_MULTIPLEX_VERSION in the IQ database. Each row in the ETL_MULTIPLEX_STATE table signifies an IQ writer selected by an ETL gridnode, which is removed automatically after each execution. In case, gridnode crashes due to unexpected error, you *must* manually clean the data in this table.

You can set the required permissions using Sybase Central:

1 Go to Sybase Central and connect to the Sybase IQ 15.0 server.

2 Expand Users & Groups and then select the user for whom you want to set the create table permission.

3 Right-click the user and select Properties.

4 Select the Authorities tab and check the Resource option to give the user permission to create database objects in the IQ database.

5 Select the Permissions tab and then select the Procedures & Functions option to see a list of all the available permissions.

6 Select sp_iqstatistics and click on the corresponding Execute column to give the user permission to execute the stored procedure in the IQ database.

7 Click OK to save the settings.

Note To support multiplex execution, you must install the SQL Anywhere 11 ODBC driver on the same machine as ETL Development and ETL Server.

You must configure the writers to be used for multiplex execution by defining them in the *IQMultiplex.ini* file.

❖ **Defining preferred writers in the *IQMultiplex.ini* file**

1 Navigate to the *etc* directory in the installation folder and use a text editor to open the *IQMultiplex.ini* file.

2 Add one section for each multiplex group you want to use. By default, the *IQMultiplex.ini* file is empty. If you do not specify anything, the grid engine uses the internal default values.

A sample section is as follows:

```
[dbsybase15+iq15m+sample]
/* This is the section name */
Enabled=true
Workload=OperationsWaiting
MinimalUpdateInterval=60
SelectWriterTimeout=6
MostIdleNode=select NAME from TEST_CUSTOMER_NODE
order by WORKLOAD asc
MostBusyNode=select NAME from TEST_CUSTOMER_NODE
order by WORKLOAD desc
```

You must provide the section name, which includes the coordinator's interface name, host name, and database name, separated by a plus (+) sign. Do not use colon (:), hash (#), or equal sign (=) characters. The other properties for each group are optional and discussed in Table 5-1.

3 Save and close the file.

Table 5-1: Multiplex group optional properties

Name	Type	Value	Description
Enabled	boolean	True (default) or False	Enable or disable this feature in the IQ server database.
Workload	text	OperationsWaiting (default)	Specify which row of the result of EXEC sp_iqstatistics should be used to take as workload. The dispatcher executes the stored procedure EXEC sp_iqstatistics to query the writer workload. The query result set returns the operation status name in the second column, and the status value in the fourth column. The dispatcher finds the row for which the second column matches the value of the Workload option you specify, and then uses the fourth column of that row as the final workload value.
MinimalUpdateInterval	integer	Greater than zero (0), default value is 6	The minimal interval in seconds, for the dispatcher to refresh writer information by querying the coordinator.

Name	Type	Value	Description
SelectWriterTimeout	integer	Greater than equal to zero (0), default value is 0	The number of seconds the dispatcher should wait, if all the writers are selected and not released. When you specify 0, the dispatcher waits indefinitely. In case of a timeout, an error is generated.

MostIdleNode	SQL	Empty by default	The first column of the first row returned by the SQL execution should be a writer node name. The dispatcher assumes the returned writer as the most idle node in the multiplex, and uses it as the target for the next table load request.
			For example, this SQL script creates a custom dispatch table:
			`DROP TABLE TEST_CUSTOMER_NODE;`
			`CREATE TABLE TEST_CUSTOMER_NODE(`
			`NAME varchar(100),`
			`WORKLOAD int /*must be integer*/`
			`);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w1',78);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w2',34);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w3',12);`
			`/* iq15w1-w3 are writers*/`
			To obtain the most idle node from the table, add this SQL query in the *IQMultiplex.ini* file:
			`Select NAME from TEST_CUSTOMER_NODE order by WORKLOAD asc`
			iq15w3 is returned as the most idle node.

Name	Type	Value	Description

MostBusyNode	SQL	Empty by default	The first column of first row returned by the SQL execution should be a writer node name. The dispatcher assumes the returned writer as the busiest node in the multiplex, and delays using it as a load table target.
			For example, this SQL script creates a custom dispatch table:
			`DROP TABLE TEST_CUSTOMER_NODE;`
			`CREATE TABLE TEST_CUSTOMER_NODE(`
			`NAME varchar(100),`
			`WORKLOAD int /*must be integer*/`
			`);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w1',78);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w2',34);`
			`INSERT INTO TEST_CUSTOMER_NODE (NAME,WORKLOAD)VALUES('iq15w3',12);`
			`/* iq15w1-w3 are writers*/`
			To obtain the most busy node from the custom dispatch table, add this SQL query in the *IQMultiplex.ini* file:
			`Select NAME from TEST_CUSTOMER_NODE order by WORKLOAD desc`
			iq15w1 is returned as the most busy node.

DB Bulk Load Sybase IQ properties list

DB Bulk Load Sybase IQ properties list identifies the connection parameters and other items you define on the Database Configuration window.

Required properties

Property	Description
Interface	Specify the method or driver to use to connect to the data source.
Host Name	Specify the host where the Sybase IQ target is running.
Destination	Select the destination table from a set of existing tables.

Property	Description
Load Stage	Specify a Data File path or pipe name. The load stage file must reside on the same machine as the IQ Server. If you use named pipes on UNIX or Linux, the ETL Server and the IQ Server must reside on the same machine. This is not a requirement for Windows. **Note** If you have selected the Use IQ Client Side Load option, provide a file path name instead of a pipe name in the Load Stage field. Client side loading is not supported using named pipes.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Shared Connection	Select this option to allow the component to share a single connection to the database with other target components that have identical connection and database parameters. A connection can be shared only between components within the same project. Components in different projects within the same job cannot share a connection Components that use the same database interface and login information, but have differing database options cannot share a connection, and generate an error when the project is executed or simulated. **Note** Connection sharing is not supported if the Use IQ Multiplex property is enabled.
Key	Select target key attributes to identify the records on Upsert or Delete operations. If no key is selected, the interface works with the primary key information, which is delivered from the DB host. An error appears if no primary key information is available.

Property	Description
Function	Select one of these load functions: • Insert (default) – load records directly into the selected target table using the specified file path or pipe name. • Upsert – update existing records and insert the new records. Existing records are replaced and not updated on an attribute level. You can use the Key property to specify the target attributes identifying the records you want to update. • Delete – delete records from the target tables based on the keys in the incoming data. You can use the Key property to specify the target attributes identifying the records you want to delete. If you have selected the Truncate option, all records are removed from the target table before loading. In this scenario, the selected functions perform as follows: • Insert and Upsert load all records to the target table directly. • Delete does not move any records, but Pre-processing and Post-processing SQL are still executed.
Truncate	Remove all records from the destination table before the load.
Use IQ Client Side Load	Add records from files located on remote host machines into the Sybase IQ table, using the LOAD TABLE statement.
Load Script	The LOAD TABLE Statement is generated at runtime based on the component settings, if this property is empty. To use a customized script, right-click the component and select Generate Load Script. The LOAD TABLE script is generated for Insert. After you generate the script, you can click Load Script and edit the script. **Note** If a custom Load Script is provided, the Function property is ignored.

Property	Description
Load Stage (Server)	Specify the server path to the data file or, leave it empty when using a pipe. If the Sybase IQ server needs to use a different path to the temporary data file than specified in the Load Stage property, enter it here.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;). **Note** If you have selected the Truncate option, all records from the destination table are deleted before the Pre Processing SQL is executed.
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Converts incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (".") as the decimal separator.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See "Database connection settings" on page 94.

Property	Description
IQ Lock Table in Exclusive Mode	Lock the target table and prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. The IQ Lock Table in Exclusive Mode option also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.
Use IQ Multiplex	Support multiplex execution by using multiple writers to load data into the IQ database.
Transactional	All work performed by the DB Bulk Load Sybase IQ component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See "Job components" on page 210 and "Enabling transactionality for projects and jobs" on page 18 for information on the "Propagate Rollback" property.

DB Bulk Load Sybase IQ and DB Space

If you use the Bulk Load Sybase IQ component, and the project or job takes a long time to execute, check the Sybase IQ console or log. If you see an "out of space" message, you must add another dbspace. The message in the IQ message file indicates which dbspace has run out of space and the minimum number of megabytes to add. If the problem occurs while you are inserting data, you may need more room in the IQ Store. If the problem occurs during queries with large sorts and merges, you may need more room in the Temporary Store.

This SQL statement adds 100MB of database space to the existing asiqdemo database on Windows:

```
CREATE DBSPACE asiqdemo2 AS
```

```
'd:\\sybase\\ASIQ-12_7\\demo\\asiqdemo2.iq'
IQ STORE
SIZE 100;
```

This SQL statement adds 200MB of temporary space to the existing asiqdemo database:

```
CREATE DBSPACE asiqdemotmp AS
'd:\\sybase\\ASIQ-12_7\\demo\\asiqdemo2.iqtmp'
IQ TEMPORARY STORE
SIZE 200 ;
```

Note For additional information about diagnosing potential memory problems, see "Resource issues" in the *Sybase IQ Troubleshooting and Recovery Guide*.

Customizing the IQ Loader data format

The IQ Loader Interface uses default values for delimiters, null handling, and character sets when writing to the data file or pipe, and when generating the LOAD TABLE script. You can specify default values in the ETL Server *INI* files as follows:

Group	Key	Values	Default	Description
iq_loader	rowdelim	Any string. '\n' for line break.	' \n '	Line delimiter
iq_loader	coldelim	Any string. '\t' for tab.	' |@#& '	Column delimiter
iq_loader	nullreplace	Any string. If empty, NULL clause is not added to the load script.	' [NULL] '	String used for NULL values
iq_loader	characterset	Any character set supported by IQ.	' ' (=auto)	Encoding used in data file

DB Data Sink Delete

DB Data Sink Delete is a Destination component that removes records from a database destination table that match the incoming values of a selected key. If there are no matching records, DB Data Sink Delete does not display an error message.

Configuring a DB Data Sink Delete component

1 Drag DB Data Sink Delete component onto the Design window.

2 In the Database Configuration window, add the connection parameters for the target database. Specify a valid interface and host name.

See the “DB Data Sink Delete properties list” on page 177 for specific field requirements.

3 Specify the table to which you want to write the transformation results. You can click the Destination Table icon to select the table, or manually enter the name of the table in the Destination Table field.

You can write the transformation results to an existing table or add a destination table based on existing ports in the project. For information on how to add a destination table, see “Adding a Destination table” on page 181.

To add a table based on an existing port structure, skip this step.

4 Click Finish.

5 In the Properties window, click the Key icon to select the columns identifying the records to remove from the Destination table.

You must select a Destination table before you can select a key, and you can select multiple key columns. This is a logical selection, not related to any underlying indexes in the database schema.

6 Specify any other optional properties in the Properties window.

❖ **Updating port structure with database changes**

- See “Updating port structure with database changes” on page 92.

❖ **Loading data at IN-port to a database or a text file**

- Right-click the DB Data Sink Delete component and select Flush Buffer.

The Flush Buffer option writes out the buffered rows to the target. All components for which a write block size has been specified, can buffer data until the write block size is reached. At any time during simulation, if there is data in the buffer, the Flush Buffer option will display along with the number of rows that are yet to be written to the target. If the buffer is empty, the option will be grayed out.

DB Data Sink Delete properties list

The following tables list the required and optional properties of the DB Data Sink Delete component.

Required properties

Property	Description
Interface	Identify the method or driver you want to use to connect to the data source.
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.
Destination Table	Select the destination table from a set of existing tables. You can also create a new Destination table based on a component's port structure. See "Adding a Destination table" on page 181 for more information.
Key	Select the columns of the destination table that identify the records to delete. You must select a Destination table before you can select a key. You can select multiple key columns.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Shared Connection	Select this option to allow the component to share a single connection to the database with other target components that have identical connection and database parameters. A connection can be shared only between components within the same project. Components in different projects within the same job cannot share a connection Components that use the same database interface and login information, but have differing database options cannot share a connection, and generate an error when the project is executed or simulated.
Write Block Size	Specify the number of records to be written to the file in a single write operation.

Property	Description
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Opening Attribute Quote	Prefix for attribute names in SQL statements.
Closing Attribute Quote	Postfix for attribute names in SQL statements.
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (“.”) as the decimal separator.
IQ Lock Table in Exclusive Mode	Lock the target table and prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. The IQ Lock Table in Exclusive Mode option also queues multiple projects that load the same table in Sybase IQ.

Property	Description
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.
Load Stage Path	Specify a data file path. The load stage file must reside on the same machine as the IQ Server. When using a Sybase IQ database, if you specify a Load Stage Path, the component uses the LOAD TABLE statement instead of using SQL statements. This leads to faster performance. **Note** You do not have to enter the Load Stage Path if client side load balancing capability is available on your IQ Server. If available, it is automatically used with the LOAD TABLE statement to add records from files located on remote host machines into the Sybase IQ table. To create a pipe, specify pipe:// as the Load Stage parameter. A pipe is not used if the Load Stage is blank. If you use named pipes on UNIX or Linux, the ETL Server and the IQ Server must reside on the same machine. This is not a requirement for Windows.
Load Stage (Server)	Specify the server path to the data file or, leave it empty when using a pipe. If the Sybase IQ server needs to use a different path to the temporary data file than specified in the Load Stage property, enter it here.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See "Database connection settings" on page 94.

Property	Description
Transactional	All work performed by the DB Data Sink Delete component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

DB Data Sink Delete demos

Sybase ETL includes several demonstrations for the DB Data Sink Delete component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Destination | DB Data Sink - Delete.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.Repository | Projects, and then select Demo DB Datasink Delete.

DB Data Sink Insert

DB Data Sink Insert is a Destination component that adds records from the IN-port to a database table. You can exclude attributes or assign default values to determine the records you insert into the table.

Configuring a DB Data Sink Insert component

1 Drag the DB Data Sink Insert component onto the Design window.

2 In the Database Configuration window, add the connection parameters for the target database.

3 See the “DB Data Sink Insert properties list” on page 182 for specific field requirements.

4 Select or enter the Destination table.

You can write to an existing table or add a table based on existing ports in the project. See “Adding a Destination table” on page 181 for additional information. To add a table based on an existing port structure, skip this step.

5 Click Finish.

6 Specify any other optional properties in the Properties window.

❖ **Updating port structure with database changes**

- See “Updating port structure with database changes” on page 92.

❖ **Loading data at IN-port to a database or a text file**

1 Right-click the DB Data Sink Insert component and select Flush Buffer.

The Flush Buffer option writes out the buffered rows to the target. All components for which a write block size has been specified can buffer data until the write block size is reached. At any time during simulation, if there is data in the buffer, the Flush Buffer option appears, along with the number of rows that are yet to be written to the target. If the buffer is empty, the option is not available.

Adding a Destination table

You can write the transformation results to an existing table or add a Destination table based on existing ports in the project. You cannot add a table based on a port structure from the Database Configuration window or Properties window. You must select the port structure in the Design window.

❖ **Writing to an existing table**

1 In the Database Configuration window, specify the table where you want to write the transformation results. You can click the Destination Table icon to select the table, or manually enter the name of the table in the Destination Table field.

2 Click Finish.

❖ **Adding a destination table based on the IN-port**

1 In the Design window, right-click the component, and select Add Destination Table from Input.

2 Name the new table. Click OK.

3 Verify that the table information is correct, and click Create.

❖ **Adding a destination table from an existing port**

1 In the Design window, right-click the component, and select Add Destination Table from Port.

2 Name the new table. Click OK.

3 Select the port whose structure you want to assign to the new table. Click Apply.

4 In the Add table window, verify if the table information is correct and click Create.

Note You can also create a Destination table with your own toolset, or select an existing table from the Properties window.

DB Data Sink Insert properties list

The following tables list the required and optional properties of the DB Data Sink Insert component.

Required properties

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Identify the data source. The options that appear on the Host Name list depend on the interface you select.
Destination Table	Select the destination table from a set of existing tables, or enter the destination table manually. You can also create a new Destination table based on a component's port structure. See "Adding a Destination table" on page 181.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.

Property	Description
Shared Connection	Select this option to allow the component to share a single connection to the database with other target components that have identical connection and database parameters. A connection can be shared only between components within the same project. Components in different projects within the same job cannot share a connection Components that use the same database interface and login information, but have differing database options cannot share a connection, and generate an error when the project is executed or simulated.
Insert Options	Determine how records will be inserted. The Include columns specify the attributes to get values assigned from the component. Unselect any attributes for which you want to apply the database defaults. In the SQL INSERT clause column, you can overwrite the value of the incoming attribute with a new one. You can use any expression allowed in the SQL language of the underlying database. SBN expressions are evaluated during initialization of the component, so the value or expression is always constant during a single execution. For example, SQL INSERT value clauses are: • Static value – 'valid' • Dynamic value (evaluated by ETL Server) – '[uDate("now")]' • Database function (evaluated by database server) – getdate()
Truncate Table	Remove all records from the destination table when initializing the transformation process.
Write Block Size	Specify the number of records to be written to the file or pipe in a single write operation.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).

Property	Description
Opening Attribute Quote	Specify a prefix for attribute names in SQL statements.
Closing Attribute Quote	Specify a postfix for attribute names in SQL statements.
Database	Identify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (".") as the decimal separator.
IQ Lock Table in Exclusive Mode	Lock the target table and prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. IQ Lock Table in Exclusive Mode also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.

Property	Description
Load Stage Path	Specify a data file path. The load stage file must reside on the same machine as the IQ Server. When using a Sybase IQ database, if you specify a Load Stage Path, the component uses the LOAD TABLE statement instead of using SQL statements. This leads to faster performance. **Note** You do not have to enter the Load Stage Path if client side load balancing capability is available on your IQ Server. If available, it is automatically used with the LOAD TABLE statement to add records from files located on remote host machines into the Sybase IQ table. To create a pipe, specify pipe:// as the Load Stage parameter. A pipe is not used if the Load Stage is blank. If you use named pipes on UNIX or Linux, the ETL Server and the IQ Server must reside on the same machine. This is not a requirement for Windows.
Load Stage (Server)	Specify the server path to the data file or, leave it empty when using a pipe. If the Sybase IQ server needs to use a different path to the temporary data file than specified in the Load Stage property, enter it here.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Transactional	All work performed by the DB Data Sink Insert component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

DB Data Sink Insert demos

Sybase ETL includes several demonstrations for the DB Data Sink Insert component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Destination | DB Data Sink - Insert.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.
Repository | Projects. Then select:

- Demo Transfer German Customers
- Demo Transfer German Products
- Demo Transfer German Sales
- Demo Transfer U.S. Customers
- Demo Transfer U.S. Products

DB Data Sink Update

DB Data Sink Update is a Destination component that updates or overwrites all records that match a selected key. This component does not insert new records. If there are no matching records, DB Data Sink Update does not display an error message.

Note If the update values violate any restrictions of the underlying table or object, such as constraints, referential integrity, or unique index definition, an error message appears. The selected Key Value attribute is independent of any existing index definitions.

Configuring a DB Data Sink Update component

1 Drag DB Data Sink Update onto the Design window.

2 On the Database Configuration window, add the connection parameters for the target database.

See the “DB Data Sink Update properties list” on page 187 for specific field requirements.

3 Specify the Destination table to which you want to write the transformation results. You can click the Destination Table icon to select the table, or manually enter the name of the table in the Destination Table field.

You can write the transformation results to an existing table or add a Destination table based on existing ports in the project. For information on how to add a destination table, see “Adding a Destination table” on page 181.

To add a table based on an existing port structure, skip this step.

4 Click Finish.

5 In the Properties window, click the Key icon to select the columns of the Destination table that identify the records to update.

You must specify a Destination table before you can select a key, and you can select multiple key columns. This is a logical selection, not related to any underlying indexes in the database schema.

6 Specify any other optional properties in the Properties window.

❖ **Updating port structure with database changes**

- See “Updating port structure with database changes” on page 92.

❖ **Loading data at IN-port to a database or a text file**

- Right-click the DB Data Sink Update component and select Flush Buffer.

The Flush Buffer option writes out the buffered rows to the target. All components for which a write block size has been specified can buffer data until the write block size is reached. At any time during simulation, if there is data in the buffer, the Flush Buffer option appears, along with the number of rows that are yet to be written to the target. If the buffer is empty, the option is not available.

DB Data Sink Update properties list

The following tables list the required and optional properties of the DB Data Sink Update component.

Required properties

Property	Description
Interface	Specify the method or driver to use to connect to the data source.

Property	Description
Host Name	Identify the data source. The options that appear on the host name list depend on the interface you select.
Destination Table	Select the destination table from a set of existing tables, or enter the destination table manually. You can also create a new Destination table based on a component's port structure. See "Adding a Destination table" on page 181.
Key	Select the columns of the Destination table that identify the records to update. You must select a Destination table before you can select a key, and you can select multiple key columns.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Shared Connection	Select this option to allow the component to share a single connection to the database with other target components that have identical connection and database parameters. A connection can be shared only between components within the same project. Components in different projects within the same job cannot share a connection Components that use the same database interface and login information, but have differing database options cannot share a connection, and generate an error when the project is executed or simulated.

Property	Description

Update Options	Select the attributes (key attributes are not listed) to include in the update. By default, all attributes are selected. Unselect the attributes to exclude from the update.
	In the SQL UPDATE SET clause column, you can overwrite the value of the incoming attribute with a new one.
	In SQL language notation the, contents of the columns are processed as:
	`UPDATE customers`
	`SET cu_createdate = '2005-02-26'`
	`WHERE ….`
	You can use any expression allowed in the SQL language of the underlying database. SBN expressions are evaluated during initialization of the component, so the value or expression is always constant during a single execution.

Write Block Size	Specify the number of records to be written to the file or pipe in a single write operation.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Opening Attribute Quote	Specify a prefix for attribute names in SQL statements.
Closing Attribute Quote	Specify a postfix for attribute names in SQL statements.
Database	Specify the database to use as data source. If you select this option, you must also select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identify the schema/owner you want to use as data source. The objects that appear are restricted accordingly and new tables are created in that schema.

Property	Description
Standardize Data Format	Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase ETL can move between systems that support different formats. Dates are converted into a format that includes the year, month, day, hour, minute, seconds, and fraction of a second: `YYYY-MM-DD hh:mm:ss.s`. For example: `2005-12-01 16:40:59.123` Numbers are converted using a period (".") as the decimal separator.
IQ Lock Table in Exclusive Mode	Lock the target table and prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. The IQ Lock Table in Exclusive Mode option also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.

Property	Description
Load Stage Path	Specify a data file path. The load stage file must reside on the same machine as the IQ Server. When using a Sybase IQ database, if you specify a Load Stage Path, the component uses the LOAD TABLE statement instead of using SQL statements. This leads to faster performance. **Note** You do not have to enter the Load Stage Path if client side load balancing capability is available on your IQ Server. If available, it is automatically used with the LOAD TABLE statement to add records from files located on remote host machines into the Sybase IQ table. To create a pipe, specify pipe:// as the Load Stage parameter. A pipe is not used if the Load Stage is blank. If you use named pipes on UNIX or Linux, the ETL Server and the IQ Server must reside on the same machine. This is not a requirement for Windows.
Load Stage (Server)	Specify the server path to the data file or, leave it empty when using a pipe. If the Sybase IQ server needs to use a different path to the temporary data file than specified in the Load Stage property, enter it here.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Transactional	All work performed by the DB Data Sink Update component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

DB Data Sink Update demos

Sybase ETL includes a demonstration for the DB Data Sink Update component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Demonstrations | Destination | DB Data Sink - Update.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.
Repository | Projects, and then select Demo DB Datasink Update.

Text Data Sink

Text Data Sink is a Destination component that writes transformation results to a text file in a delimited or fixed-length format.

Configuring a Text Data Sink component

1 Drag the Text Data Sink component onto the Design window.

The OUT-port of Text Data Sink should link to the IN-port of the component that provides the inbound data when you add the component to the project.

See “Text Data Sink properties list” on page 194 for specific field requirements. Also see:

- “Exporting and importing file definitions” on page 193 – export and import options on the Component window let you save the file properties to a definition file, and reuse them for other components.
- “Modifying the port structure (delimited files)” on page 194 – column values for delimited files reflect the current IN-port structure. You can assign a port structure based on another port or re-create the current port structure.
- “Working with fixed-length files” on page 194 – if you work with fixed-length file types, you must create the columns, and provide the position parameters for each column.

2 Click Save.

3 Specify any other optional properties in the Properties window.

Note If there is an adjacent component available in the Design window, Sybase ETL automatically creates a link between the Text Data Sink IN-port with the OUT-port of that component. For delimited files, this link provides the initial port structure you see when the window opens.

If not, you may need to close the Design window and connect the Text Data Sink IN-port with the OUT-port of an adjacent component.

Note Text Data Sink does not impact the simulation sequence.

❖ **Loading data at IN-port to a database or a text file**

- Right-click the Text Data Sink component and select Flush Buffer.

The Flush Buffer option writes out the buffered rows to the target. All components for which a write block size has been specified can buffer data until the write block size is reached. At any time during simulation, if there is data in the buffer, the Flush Buffer option appears, along with the number of rows. If the buffer is empty, the option is not available.

Exporting and importing file definitions

Export and Import options on the Component window let you save file properties to a definition file, and reuse them for other components. Export saves the component properties to a definition file. Import loads a definition file you created with the Export command.

❖ **Exporting file definitions**

1 To open the Component window, double-click Text Data Sink.

2 Click Properties | Export.

3 Select the definition file you want to use.

❖ **Importing file definitions**

1 To open the Component window, double-click Text Data Sink.

2 Click Properties | Import.

3 Select the definition file you want to use.

Modifying the port structure (delimited files)

Column values on the Text Data Sink Components window reflect the current IN-port structure. You can assign a new port structure or re-create the current port structure.

❖ **Assigning a new port structure**

1 To open the Component window, double-click Text Data Sink.

2 In the Column Names pane, click the Assign Port Structure icon.

3 Select the port whose structure you want to assign.

❖ **Regenerating column definitions**

1 To open the Component window, double-click Text Data Sink.

2 In the Column Names pane, right-click the Regenerate the column definition icon.

Working with fixed-length files

For fixed length file types, you must create the columns and provide the position parameters for each column.

❖ **Adding columns to the output**

1 To open the Component window, double-click Text Data Sink.

2 In the Column Names pane, click the Insert a New Attribute icon available. You can edit the name of the generated column.

❖ **Removing columns from the output**

1 To open the Component window, double-click Text Data Sink.

2 Select the column and click the Remove an attribute icon.

Text Data Sink properties list

The following tables list the required and optional properties of the Text Data Sink component.

Required properties

Property	Description
Text Destination	Specify the output file. Text Data Sink prompts you for the destination file when you add a component to a project. To specify a destination file, click the Destination File icon in the Properties window, and select an existing file, or type the full path and file name to create one during project execution.
Columns	Define columns for data in the source file. If there is a property value defined, the Columns value reflects the port structure or attribute values you defined on the Component window.

Optional properties

Property	Description
Row Delimiter	Specify how each row is delimited: • Position (fixed-line position) • LF (line feed) • CR (carriage return) • CRLF (carriage return followed by a line feed) Alternatively, you can enter a different delimiter.
Row Length	If you have selected Position as the Row Delimiter, specify the number of characters in each fixed row.
Column Delimiter	Specify how columns are delimited: • Position (fixed-column positions) • Tab • Comma • Semicolon Alternatively, you can enter a different delimiter.
Column Quote	Specify how you want the values in the output file to be quoted (delimited files only): • None • Single quote • Double quote Alternatively, enter a different quote character or string

Property	Description
Fixed by Bytes	Specify how to interpret the values provided for line length, column start, and column end: • Not selected (default value) – values are interpreted as number of characters. • Selected – values are interpreted as number of bytes. For example, suppose your source file includes binary: **0x61 62 63 d6 d0 ce c4 61 62 63 64 65** and has these characteristics: • File Type – fixed (variable line) • Encoding – GB2312 • Row delimiter – '\n' • Column definition – column1: 1 – 7; column 2: 9 –10 If you select Fixed by Bytes: • Column 1 displays the first 7 bytes, the binary of which are **0x61 62 63 d6 d0 ce c4**. • Column 2 displays the 9th and 10th bytes, the binary of which are **0x62 63**. If you do not select Fixed by Bytes: • Column 1 displays the first 7 characters, the binary of which are **0x61 62 63 d6d0 cec4 61 62**. • Column 2 displays the next 2 characters, the binary of which are **0x64 65.** **Note** 0xd6d0, c4c4 represents 2 Chinese characters in GB2312.
Encoding	Set the current character encoding.
Append Column Delimiter	Select if you want the column delimiter to be appended to the end of a row. Selecting this option enhances the performance while loading output files into IQ.
Column Header	Write the column names to the file.
Header	Create a report header to write to the file. Text Data Sink writes the header before it writes the incoming data. Enter the header text; you can use Square Bracket Notation, if you want.

Property	Description
Append Data	Append incoming data to the destination file. If you do not set this value, Text Data Sink overwrites any existing data in the destination file.
Write Block Size	Specify the number of records that Sybase ETL writes to the file in a single write operation.
Transactional	All work performed by the Text Data Sink component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

Text Data Sink demos

Sybase ETL includes several demonstrations for the Text Data Sink component. These demos are available as flash demos and as sample projects in the demo repository.

To run the flash demo, select Help | Component Demonstrations | Destination | Text Data Sink.

To access the sample projects, in the Navigator, select Repository | TRANSFORMER.transformer.
Repository | Projects. Then select:

- Demo XML via SQL Data Provider
- Demo Text Data Sink Delimited/Fixed

Loader components

Loader components help to load data from a source database or a file into the IQ database, without performing any transformation.

Component	Description
IQ Loader File via Load Table	Use this component to load data from a file into a target IQ database using the LOAD TABLE statement.

Component	Description
IQ Loader DB via Insert Location	Use this component to load data from source database into a target IQ database using the INSERT LOCATION statement.

IQ Loader File via Load Table

The IQ Loader File via Load Table component loads data from a file into a target IQ database using an automatically generated LOAD TABLE statement.

This self-contained component operates as a data source when it reads from source files and a data sink when it writes to a Sybase IQ database. It does not have IN and OUT-ports, therefore, there is no need to create one component for performing the read from the source file and another component for invoking Load Table in Sybase IQ. ETL automatically generates Load Table statements that extract data from a delimited text file and loads the data to Sybase IQ.

Configuring IQ Loader File via Load Table component

1 Drag IQ Loader File via Load Table into the Design window.

2 In the Database Configuration window, add the connection parameters for the target IQ database. See “IQ Loader File via Load Table properties list” on page 199 for specific field requirements.

3 Select or enter the Destination table.

4 Click Finish.

5 Specify any other optional properties in the Properties window.

Working with the Text Source property window

The Text Source property window lets you define the structural properties of data in the source file:

- File Content pane – displays the contents of the source file.

- Properties pane – displays the file description properties.

  Note When you select a source file, the file path displayed in the Text Source field is the path of the machine where ETL Development is running. If your grid engine is running on another machine, select a local file, save, and close the window. Then, reopen the window and replace this path with the file path of the machine on which your grid engine is running.

- Preview pane – displays a tabular view of the data in the source file, based on the currently selected properties.

Enabling client-side load support

You can use the IQ Loader File via Load Table component to load data from files that are located on remote host machines into the Sybase IQ table. See "Enabling client-side load support" on page 165.

Configuring IQ multiple writers for loading data

To enable support for multiplex execution by using multiple writers for loading data to IQ, you must make some additional configurations. For detailed configuration steps, see "Configuring multiple writers for loading data" on page 166.

IQ Loader File via Load Table properties list

The following tables list the required and optional properties of the IQ Loader File via Load Table component.

Required properties

Property	Description
Interface	Specify the method or driver with which to connect to the target IQ database. The supported interfaces are Sybase and ODBC.
Host Name	Specify the host where the Sybase IQ target is running.

Optional properties

Property	Description
User and Password	Identify an authorized database user, and protect the database against unauthorized access.
Destination	Select the destination table from a set of existing tables.
Key	Select target key attributes to identify the records on Upsert or Delete operations. If no key is selected, the interface works with the primary key information, which is delivered from the DB host. An error is displayed if no primary key information is available.
Function	Select one of these load functions: • Insert – to load records directly from the file into the selected target table. • Upsert – to update the existing records and insert the new records. When you select Upsert, the existing records are replaced and not updated on an attribute level. You can use the Key property to specify the target attributes to identify the records you want to update. If you have selected the Truncate option, all records are removed from the target table before loading. The Insert and Upsert function loads all records to the target table directly.
Use Binary Load File	Select to load data from an IQ binary load file. **Note** If the Use Binary Load File property is selected, you can only define the source file path in the Text Source property. No other properties can be specified.
Text Source	Identify the text file you want to use as the data source. From the Properties window, click the Text Source icon, select the file, and specify the format. See “Working with the Text Source property window” on page 198.
Use IQ Client Side Load	Bulk load data from files located on remote host machines into a target IQ database, using the LOAD TABLE statement.

Property	Description
Row Delimiter	Specify how each row is delimited: • LF (line feed) • CR (carriage return) • CRLF (carriage return followed by a line feed) Alternatively, you can enter a different delimiter.
Column Delimiter	Specify how columns are delimited: • Tab • Comma • Semicolon • Pipe Alternatively, you can enter a different delimiter.
Load Script	The LOAD TABLE statement is generated at runtime based on the component settings, if this property is empty. To use a customized script, right-click the component and select Generate Load Script. The LOAD TABLE script is generated for Insert. After you generate the script, you can click Load Script and edit the script. **Note** If a custom Load Script is provided, the Function property is ignored.
Truncate	Remove all records from the destination table before the load.
Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;). **Note** If you have selected the Truncate option, all records from the destination table are deleted before the Pre Processing SQL is executed.
Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).

Property	Description
Database	Specify the database to use as data target. The database is used together with the specified user name, password, and host name.
Schema	Specify an owner to filter the table catalog.
Database Options	Set options that override performance defaults and control the behavior of some transactions. See "Database connection settings" on page 94.
Null Indicator	Specify the string that represents null values in the source file.
Skip Rows	Specify the number of rows to skip at the beginning of the input file for a load. The default is 0.
Parallel format	Allow the LOAD TABLE command to run in parallel. To use this option, all columns, including the last, must be delimited by a single ASCII character.
Strip	Strip trailing blanks from values before they are inserted. This applies only to variable-length nonbinary data.
Byte Order	Specify the byte ordering during reads. This option applies to all binary input fields. If none are defined, this option is ignored. Sybase ETL always reads binary data in the format native to the machine it is running on (default is NATIVE). You can also specify: • HIGH when multibyte quantities have the high-order byte first. • LOW when multibyte quantities have the low-order byte first.
Block Size	Specify the default size, in bytes, in which input should be read.
Limit	Specify the maximum number of rows you want to insert into the table. The default is 0 for no limit.

Property	Description
ON File Error	Specify the action Sybase IQ should take when an input file cannot be opened, either because it does not exist or because you have incorrect permissions to read the file. For all other reasons or errors, it aborts the entire insertion. You can specify one of the following options: • ROLLBACK (the default) – aborts the entire transaction. • FINISH – finishes the insertions already completed and ends the load operation. • CONTINUE – returns an error but only skips the file to continue the load operation. You cannot use this option with partial-width inserts.
Word Skip	Allow the load to continue when it encounters data longer than the limit specified when the word index was created.
IQ Lock Table in Exclusive Mode	Lock the target table to prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. The IQ Lock Table in Exclusive Mode option also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.
Use IQ Multiplex	Support multiplex execution by using multiple writers to load data into the IQ database.

Property	Description
Transactional	All work performed by the IQ Loader File via Load Table component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See “Job components” on page 210 and “Enabling transactionality for projects and jobs” on page 18 for information on the “Propagate Rollback” property.

IQ Loader DB via Insert Location

The IQ Loader DB via Insert Location component loads data from the source database into the target IQ database using the Insert Location statement.

This self-contained component operates as a data source when it reads from a source database and a data sink when it writes to a Sybase IQ database. It does not have IN-ports and OUT-ports, therefore, there is no need to create one component for performing the read from the source database and another component for invoking Insert Location in Sybase IQ. ETL automatically generates Insert Location statements to transfer data from the source database to Sybase IQ. Insert Location:

- Allows you to move columns from Sybase IQ versions earlier than 12.0 to version 12.0 and later.
- Provides optimized load to Sybase IQ from Adaptive Server Enterprise or Sybase IQ.
- Also works with Sybase Enterprise Connect™ Data Access (ECDA) to load Sybase IQ from Oracle, IBM DB2, and Microsoft SQL Server. ETL supports Sybase ECDA 15.0 with IBM DB2 9.1, Oracle 10g, and Microsoft SQL Server 2005.

  For Sybase ECDA documentation, see the Sybase Product Manuals Web site at http://www.sybase.com/support/manuals.

Note Sybase is the only interface supported by the IQ Loader DB via Insert Location component.

Configuring the IQ Loader DB via Insert Location component

1. Drag the IQ Loader DB via Insert Location component into the Design window.
2. Enter IQ database connection properties for the destination database.
   - Host – select the IQ host.
   - User – enter an authorized database user name.
   - Password – enter the password for the database user.
   - Database – select the database to use as the destination database.
   - Schema – select schema/owner to restrict the objects that appear and create new tables in that schema.
   - Click Processing to enter preprocessing and postprocessing SQL on the IQ destination database.
   - Click Logon to view the list of available tables.
   - Click Next.
3. Enter connection information for the source database and select the tables to transfer.
   - Select "Use remote server definition for accessing source database" to get data and metadata from the source. Select this option only if you have used the Create Server command to define the source server as a remote server on the destination IQ database. If you do not select this option, you are directly connected to the source data using the configuration information provided in the *.INI* or *interfaces* file.
   - Host – select the data source.
   - Database – select the database you want to use.
   - Schema – select schema/owner to restrict the objects displayed and create new tables in that schema.
   - Click Processing to enter preprocessing and postprocessing SQL on the source database.
   - Select "Create Target Tables" to create the destination tables if they do not exist.
   - Select "Continue on Error" if you want processing to continue even if an error occurs when loading data into a database.

- Select the Encrypted Password option to transmit the password in encrypted format.

  Note When used as a remote server, Sybase IQ does not support password encryption.

- Select the Use IQ Multiplex option to support multiplex execution by using multiple writers for loading data to IQ. Select this option if more than one table is being migrated to the IQ database.
- Select the Lock Table option to lock the target table in Exclusive mode and prevent it from being updated by concurrent transactions. If selected, no other transaction can execute queries or perform any updates against the locked table. The Lock Table option also queues multiple projects that load the same table in Sybase IQ.

  If you select this option, you must also specify the maximum blocking time that the project should wait before acquiring the lock.
- Enter a network packet size in the Packet Size field.
- Enter a value for Limit Rows.
- Specify the number of rows to skip at the beginning of the input tables for a load in the Skip Rows field.
- Click Logon to view the list of available tables for the specified database. By default, each table is selected for transfer. Unselect the Transfer option for tables you do not want to transfer. You can also choose one or multiple table rows, right-click and select Exclude. To include a table for transfer, right-click and select Transfer.

  Alternatively:

  - Click the Exclude all objects from transfer icon to exclude all tables.
  - Click the Include all objects in transfer icon to include all tables.
- Click Next.

4 Verify the source tables. They should be in the following format:

```
source_schema.source_table
```

5 Select the destination tables. There should be a one-to-one mapping (one source for each destination) between sources and destinations.

6 Click “Truncate Destination” to delete all existing data rows from the destination table.

7 Click Next.

8 Review the load configuration summary. Click Finish.

Setting locales for *Insert Location* statements

When you execute Insert Location statements, Sybase IQ loads the localization information needed to determine language, collation sequence, character set, and date/time format. If your database uses a non default locale for your platform, you must set an environment variable on your local client to ensure that Sybase IQ loads the correct information.

If you set the LC_ALL environment variable, Sybase IQ uses its value as the locale name. If you do not set LC_ALL, Sybase IQ uses the value of the LANG environment variable. If neither variable is set, Sybase IQ uses the default entry in the locales file. For an example, see “Setting locales” in Chapter 11, “International Languages and Character Sets” in the *Sybase IQ 12.7 System Administration Guide*.

Configuring IQ multiple writers for loading data

To enable support for multiplex execution by using multiple writers for loading data to IQ, you must make some additional configurations. For detailed configuration steps, see “Configuring multiple writers for loading data” on page 166.

IQ Loader DB via Insert Location properties list

IQ Loader DB via Insert Location properties list identifies the connection parameters and other items you must define on the IQ Loader DB via Insert Location component window.

Required properties

Property	Description
IQ Host Name	Specify the host where the Sybase IQ target is running.
IQ User	Specify an authorized IQ user to protect the database against unauthorized access.
IQ Password	Specify a password to protect the database against unauthorized access.

Property	Description
Source Host Name	Specify the data source.
Source Database	Specify the source database.
Source Transfer List	Specify the schema qualified source table name and the target table name. In the target truncate column, specify 1 to truncate the target table, otherwise enter 0.

Optional properties

Property	Description
IQ Database	Specify the IQ destination database
IQ Schema	Specify the IQ schema/owner to restrict the objects displayed and create new tables in that schema.
IQ Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
IQ Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Use Remote Definition	Select this option only if you have defined the source server as your remote server on the destination IQ database using the Create Server command. If you do not select this option, you are directly connected to the source data using the configuration information provided in the *.INI* or *interfaces* file.
Source Schema	Specify the schema/owner to restrict the objects displayed.
Source Pre Processing SQL	Create a script that runs during component initialization. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Source Post Processing SQL	Create a script that runs after all components execute. Scripts can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).

Property	Description
Function	Select one of these load functions: • Insert – to load records directly from the source into the selected target table. • Upsert – to update the existing records and insert the new records. When you select Upsert, the existing records are replaced and not updated on an attribute level. The table you are working with must have a predefined primary key. If you have selected the Truncate option, all records are removed from the target table before loading. The Insert and Upsert function loads all records to the target table directly.
Create Target Tables	Create the destination tables if they do not exist.
Continue on Error	Continue execution even if an error occurs when loading data into the database.
Limit Rows	Specify the maximum number of rows to insert into the table. The default is 0 for no limit.
Skip Rows	Specify the number of rows to skip at the beginning of the input tables for a load. The default is 0.
Encrypted Password	Transmit the password in encrypted format.
Packet Size	Specify the network packet size.
Load Script	The Insert Location statements are generated at runtime based on the component settings, if this property is empty. To use a customized script, right-click the component and select Generate Load Script. The Insert Location script is generated for Insert. After you generate the script, you can click Load Script and edit the script. **Note** If a custom Load Script is provided, the Function property is ignored.
Use IQ Multiplex	Support multiplex execution by using multiple writers to load data into the IQ database.

Property	Description
IQ Lock Table in Exclusive Mode	Lock the target table to prevent it from being updated by concurrent transactions. When an exclusive table lock is applied, no other transaction can execute queries or perform any updates against the locked table. The IQ Lock Table in Exclusive Mode option also queues multiple projects that load the same table in Sybase IQ.
Wait Time for IQ Lock Table	Specify the maximum blocking time that the project should wait before acquiring an Exclusive lock. Specify the time argument in the format hh:nn:ss.sss. If you do not enter a time argument, the server waits indefinitely until an Exclusive lock is available or an interruption occurs. When you specify "00:00:00.000" as the time argument, an Exclusive lock is acquired as soon as the project starts.
Transactional	All work performed by the IQ Loader DB via Insert Location component, including pre-SQL and post-SQL, is done in a single database transaction that is committed when the project finishes normally. Select this option to roll back the transaction, if this component encounters an error. See "Job components" on page 210 and "Enabling transactionality for projects and jobs" on page 18 for information on the "Propagate Rollback" property.

Job components

Job components control job execution.

Component	Description
Start	Represents the beginning of a job. Start is the first component you add to any job.
Project	Identifies the project you want to run in the job. Use this component to run an individual project in a job.

Component	Description
Synchronizer	Use this component to control the flow of the job depending on the status of previously executed projects. You can define each project as critical or noncritical. Critical project failures cause the Synchronizer to signal failure.
Multi-Project	Provides a visual representation of the project groups within a job. Multi-Project combines the properties of the Project and Synchronizer components. Use this component when your job consists of a large number of independent projects that can be executed in any order.
Finish	Use this component to mark the successful end of a job.
Error	Use this component to mark the end of a failed job.

Start

Start is the first component you add to any job. To add this component to a job, drag it from the Component Store onto the Design window.

Note You can connect multiple Project components, Multi-Project components, or both, to the Start component.

Project

The Project component identifies the project you want to run in the job. Use this component to run an individual project in a job.

❖ **Adding and configuring the Project component**

1 To add a Project component to a job, drag it from the Component store onto the Design window.

2 Connect the Project component with its adjacent components.

3 Double-click the component and select a project to execute.

Required properties

Property	Description
Project Name	Select the project you want to add.

Optional properties

Property	Description
Continue on DB Write Errors	Continue project execution even if an error occurs on loading data into a database via a DB Data Sink component. If errors are ignored because of the selection of this property, the project status is 'failed'. Combined with the Reject Log this option lets you “post-process” rejected records.

Project component demos

See the sample jobs in the DemoRepository. To run the sample jobs:

1 In the Navigator, click *Repository* | *TRANSFORMER.transformer.Repository* | *Jobs*.

2 Select:

- Demo Transfer all German Data
- Demo Transfer U.S. Sales on an incremental basis

Synchronizer

Synchronizer controls the flow of the job execution.

Use the Synchronizer component to control the flow of the job depending on the status of previously executed projects. You can define each project as critical or noncritical. Critical project failures cause the job flow to follow the branch at the Error port of the Synchronizer.

The Success and Error ports of the Synchronizer component can be connected to any of:

- Multiple Project components
- Multiple Multi-Project components

- Multiple Project and Multi-Project components
- Single Finish component or Error component

❖ **Configuring the Synchronizer component**

1. Add the component to the job and connect it with all the projects that signal execution status to this component.
2. (Optional) In the Property window:
   - Click the Synchronize Options icon to select critical projects.
   - Select Commit Intermediate Work if you want all tasks performed by the preceeding projects to be committed as soon as the projects have completed successfully.
   - Select Propagate Rollback to enable transactionality for the preceeding projects. If selected, data is committed at the end of the write operation for a successful execution, and rolled back for an unsuccessful execution.

Synchronizer component demos

See the sample jobs in the DemoRepository. To run the sample jobs:

1. In the Navigator, click *Repository* | *TRANSFORMER.transformer.Repository* | *Jobs*.
2. Select Demo Transfer all German Data.

Multi-Project

The Multi-Project component provides a visual representation of the project groups within a job. Multi-Project combines the properties of the Project and Synchronizer components. The Success and Error ports of the Multi-Project component can be connected to any of:

- Multiple Project components
- Multiple Multi-Project components
- Multiple Project and Multi-Project components
- Single Finish component or Error component

Use this component when your job consists of a large number of independent projects that can be executed in any order, and even in parallel, when used in multi-engine jobs.

❖ **Configuring the Multi-Project component**

1 Add the component to the job and connect it to its adjacent component.

2 In the Properties window:

- Click the Projects Execution icon to select the projects to execute.
- (Optional) Select the Commit Intermediate Work icon if you want all tasks performed by the included projects to be committed as soon as the projects have completed successfully.
- (Optional) Select "Propagate Rollback" to enable transactionality for the contained projects. If selected, data is committed at the end of the write operation for a successful execution, and rolled back for an unsuccessful execution.

3 To add projects to the group, right-click the Project name in the Navigator and select Add Projects.

4 To remove projects from the group, select the projects in the Navigator and and select Remove Projects.

Note A project group can contain only one instance of a project. Do not add a project multiple times.

The options available for project execution are:

- Continue on Error – this option corresponds to the Continue on DB Write Errors property of a Project component. If you select this option, project execution continues even if an error occurs when loading data into a database.
- Critical – define each single project as critical or noncritical. The failure of a critical project causes the Multi-Project component to signal failure.

Multi-Project component demos

See the sample jobs in the DemoRepository. To run the sample jobs:

1 In the Navigator, click Repository | TRANSFORMER.transformer.Repository | Jobs.

2 Select Demo Transfer all U.S. Data.

Finish

Finish visually represents the end of a successful job execution. Use this component to mark the successful end of a job. You can connect the Finish component to these job components: Synchronize, Project, or Multi-Project.

Finish component demos

See the sample jobs in the DemoRepository. To run the sample jobs:

1 In the Navigator, click Repository | TRANSFORMER.transformer.Repository | Jobs.

2 Select:

- Demo Transfer all German Data
- Demo Transfer all U.S. Data
- Demo Transfer U.S. Sales on an incremental basis

Error

The Error component visually represents the end of a failed job execution. Use it to mark the end of a failed job. You can connect the Error component to these job components: Synchronize, Project or Multi-Project.

Error component demos

See the sample jobs in the DemoRepository. To run the sample jobs:

1 In the Navigator, click Repository | TRANSFORMER.transformer.Repository | Jobs.

2 Select:

- Demo Transfer all German Data

- Demo Transfer all U.S. Data

CHAPTER 6

Sybase ETL Server

Sybase ETL Server is a scalable and distributed grid engine that connects to data sources, and extracts and loads data to data targets using transformation flows designed using Sybase ETL Development. Sybase ETL Server uses User Datagram Protocol (UDP) broadcasts to inform other servers about urgent events, such as starting and stopping, as well as system failure or crash.

The default port for communication which you can change in the *INI* file, or by using the command line, is 5124.

All communication between servers is done over TCP/IP on the same port.

Note Verify there are no firewalls blocking this port and the port is not in use. You can also change the port on all server installations to a different number, if necessary.

Starting and stopping Sybase ETL Server

This section describes how to start and stop Sybase ETL Server.

Starting Sybase ETL Server

At a command prompt, enter:

On Windows:

```
GridNode
GridNode --port 5500
```

On Linux and UNIX:

```
GridNode.sh
GridNode.sh --port 5500
```

Starting Sybase ETL Server as a Windows system service

You can install and run Sybase ETL Server as a Windows system service. To run it as a system service independently of the Windows GUI, start the ETL Server after system start-up using a system user account.

Note You must have administrator privileges to install, remove, start, and stop a system service.

To install the server as a Windows system service, at the command prompt, enter:

```
GridNode.exe --install [additional parameters]
```

To remove the service, at the command prompt, enter:

```
GridNode.exe --remove
```

When you run ETL Server as a Windows service, basic events (failures, success messages, and so on) are written to the Windows event log.

Stopping Sybase ETL Server

You can stop a server from the console if it is a local or remote process. All currently running projects complete execution before the server stops. To stop ETL Server, at the command prompt, enter:

On Windows:

```
GridNode --shutdown
GridNode --shutdown --server [remotehost] --port [port]
```

On Linux and UNIX:

```
GridNode.sh --shutdown
GridNode.sh --shutdown --server[remotehost] --port [port]
```

Note To stop a grid engine running on a specified server and port, you must provide the server name and port number. If you do not provide the server name and port number, the local grid engine at the default port is stopped.

Command line parameters

This section describes all Sybase ETL Server command line parameters.

To display an overview of the available parameters, enter `GridNode --help`, or `GridNode -h` at the command prompt. The console output shows you the long and short forms for each parameter, for example:

```
--version,  -V    Displays version information
```

Note The full parameter name is always prefixed by two minus signs, whereas the short form has only one.

Table 6-1: Sybase ETL Server command line parameters

Command	UNIX	Windows	Description
install or inst	No	Yes	Installs the application as a Unix daemon or Windows service.
remove or rm	No	Yes	Removes system service start.
setoptions or so	No	Yes	Sets the command line options to be used when running as a Windows service.

Command	UNIX	Windows	Description
getoptions or go	No	Yes	Prints the command line options to be used when running as a Windows service.
background or bg	Yes	No	Sets background processes that do not overuse system resources.
no_pidfile or nopid	Yes	No	Sets the file to which the server records the daemon process ID.
console or con	Yes	Yes	Writes detailed error information and trace messages to the console.
diagnosis or diag	Yes	Yes	Lists application environment.
tracelevel or tl	Yes	Yes	Sets trace level for debugging from 0 (no trace) to 5 (very verbose).
server or s	Yes	Yes	Identifies the remote server to be used.
port or p	Yes	Yes	Identifies the port number to operate on.
version or V	Yes	Yes	Displays the application version information.
help or h	Yes	Yes	Displays help information.
licenses or ll	Yes	Yes	Identifies the short information about available licenses and their status.
nodelist or nl	Yes	Yes	Lists all known peer nodes.
shutdown or sh	Yes	Yes	Shuts down the node.
nodename or n	Yes	Yes	Sets a node name.
Command line parameters for executing projects and jobs			
dbhost *host*	Yes	Yes	Repository database host name or data source name (DSN).
dbinterface *interface*	Yes	Yes	Repository database interface.
dbdatabase *database*	Yes	Yes	Repository database name.
dbschema *schema*	Yes	Yes	Repository database schema.
dbuser *user*	Yes	Yes	Repository database user.
dbpassword *encrypted password*	Yes	Yes	Repository database password.
client *client*	Yes	Yes	Repository client name.
user *user*	Yes	Yes	Repository client user.
password *encrypted password*	Yes	Yes	Repository client password.
perflog [*args*]	Yes	Yes	Set performance log level to 0 or 1.
project [*name* | *ID*]	Yes	Yes	Execute a project by specifying the project name or ID.
job [*name* | *ID*]	Yes	Yes	Execute a job by specifying the project name or ID.
paramset [*name* | *ID*]	Yes	Yes	Specify a parameter set by name or ID for a project or job.

Command	UNIX	Windows	Description
encrypt *password*	Yes	Yes	Encrypts a password, and displays the encrypted value. Use encrypt to generate the encrypted passwords you must use with dbpassword and password.
ping *host:port*	Yes	Yes	Checks if ETL Server is running at the host and port you specify.
env "*variable1=value*; *variable2=value*; ...;*variableN=value*"	Yes	Yes	Specify additional environment variables to run with ETL Server. Use semicolons to separate multiple environment variables, and enclose the entire string of variables in double quotes.
repcdcinstancename or ri	Yes	Yes	Specify the Replication CDC Service Name.

Using ETL Server to execute projects and jobs

Sybase ETL Server can perform execution of projects and jobs on all supported platforms, using the command line parameters listed in Table 6-1 on page 219. To execute projects and jobs, use this syntax:

```
GridNode --project PROJ-1234-5678 --dbinterface dbodbc --dbhost etl_comp
--client transformer--user TRANSFORMER --password 1234ABCD
```

where project ID is "PROJ-1234-5678," the database interface is "dbodbc," host name is "etl_comp," client is "transformer," user is "TRANSFORMER," and the encrypted version of the password is "1234ABCD".

You can also use these command line parameters to execute projects and jobs:

- project – to specify projects and parameter sets by name. You can also specify jobs by name. Specifying names is easier than entering a complex project, job, or parameter set ID. This example uses the project name "LoadCustomers" and the parameter set name "myparams":

```
GridNode --project LoadCustomers --dbinterface dbodbc --dbhost etl_com
--client transformer --user TRANSFORMER --paramset myparams --password 1234ABCD
```

- encrypt – to generate an encrypted password. You must use encrypted passwords with dbpassword and password. To encrypt "mypassword", enter:

  ```
  Gridnode --encrypt mypassword
  ```

  ETL Server generates and displays the encrypted password.

- ping – to verify whether ETL Server is running on a particular host and port. To verify whether ETL Server is running on the default port on "localhost," enter:

  ```
  Gridnode --ping localhost
  ```

  If ETL Server is running, you see this message:

  ```
  localhost is alive!
  ```

  You see an error message if ETL Server is not running on the host and port you specify.

- env – to specify environment variables for projects and jobs. You can access the values for these variables at runtime with the uGetEnv function. This example uses the "LoadCustomers" project with the environment variables INPUT_FILE and OUTPUT_FILE:

```
GridNode --project LoadCustomers --dbinterface dbodbc --dbhost etl_com
--client transformer --user TRANSFORMER --paramset myparams --password
1234ABCD --env "INPUT_FILE=input.txt;
OUTPUT_FILE=output.txt"
```

Note When you enter a command, type in the commands, parameters, and values in a continuous line. The examples in this section use multiple lines only for clarity.

In Sybase ETL versions earlier than 4.5, job and project execution was performed by the ProcessQ application, which was distributed with ETL Server and used internally only by Windows. ProcessQ is now deprecated.

Executing multiple projects concurrently

To execute multiple projects concurrently on a single or multiple grid engines, or on the same remote grid engine, specify in the *Default.ini* file the maximum number of projects to execute concurrently.

Warning! Multiple users concurrently executing projects or jobs against the same remote engine may encounter problems when running projects that access the same resources such as database tables and files.

1 Navigate to the *etc* directory in the installation folder and use a text editor to open the *Default.ini* file.

2 In the Runtime section, set *MAXPROJECTS* to the number of projects that you want to execute concurrently on a given grid engine. For example:

```
MAXPROJECTS= 3
```

If you set *MAXPROJECTS* to 0 or a negative number, there is no limit to the number of projects that can be concurrently executed on the grid engine.

By default, the maximum number of projects that can be executed at one time is 10.

Note The grid engine does not incorporate a locking mechanism for multiple users running projects against the same remote grid node. The source grid engine queries the target engine for the number of projects currently running and the configured maximum number of concurrently running projects. If the number of currently running projects is fewer than the maximum, the source engine executes the project on the target engine. If two source grid engines are executing projects or jobs simultaneously, the number of projects running on a grid engine may exceed the configured maximum.

INI file settings

The *INI* files that include the settings for Sybase ETL Server are available in the *etc* folder of the installation directory.

Default.ini

Group	Key	Values	Default	Description
Network	proxy	host:port explorer	explorer	Sets the proxy for Internet access. You can fine-tune the proxy for a certain protocol by using “http_proxy”, “https_proxy”, “ftp_proxy”, or “ftps_proxy”. The proxy value “explorer” takes the system proxy in Windows environments.
Network	timeout	1 – 2147483 seconds	600 seconds	Sets a timeout value for FTP connections.
Language	Default	English_USA	English_USA	Tunes the behavior of the application based on the selected language and country.
Logging	Console	1/0	0	Sends log information to the console.
Logging	LogFile	1/0	1	Sends log information to the *system.log* file.
Logging	Tracelevel	0 – 5	0	Provides varying degrees of information for debugging. Level 0 provides the least information and level 5 provides the most detailed information. During normal execution, set this value to 0 to minimize performance impact.
Logging	Flushtime	1 – n	1	Indicates seconds between the internal log flashes.

Group	Key	Values	Default	Description

SMTP	Server	*<name or IP address of smtp server>*	〈empty〉	Sets the values for the SMTP server for alerting.
				The syntax for specifying a Server URL for the SMTP server is:
				`<protocol>://`
				`[user[:password]@]`
				`host[:port]`
				where:
				• *protocol* is smtp or secure smtp (smtps).
				• *user* is the smtp username.
				• *password* is the smtp password.
				• *host* is the hostname or the IP address.
				• *port* is the listener port number.
				Note Except for protocol, ServerURL can be substituted by using the single INI file keys, such as server, unsername, password, host, and port.
				See "uSMTP" on page 279.

	Port	*<port of the smtp listener>*	25	
	Sender	*<sender as it appears in the mail inbox>*	Sybase ETL <noreply@ localhost>	
	Username	*<login name for smtp server>*	〈empty〉	
	Password	*<encrypted password for smtp server>* **Note** Encrypt the password using GridNode --encrypt.	〈empty〉	
	Retry Count	*<positive integer describing how often ETL tries to send the mail>*	0	
	Retry Interval	*<positive integer describing the numbers of seconds between each retry>*	5	
	Server URL	*<default server URL>*	<none>	
	Recipients	*<comma separated list of default recipients>*	<none>	
	Subject	*<your default e-mail subject>*	<none>	

Group	Key	Values	Default	Description
Runtime	Runtime	Keep_History_Days = 5 MaxProjects = 10	Keep_History_Days = 5 MaxProjects = 10	Indicates days after which entries in the execution history expires.
Path	Userdata	<*startup configuration file*>	*Userdata.conf*	Specifies the configuration file to be used when ETL starts. ETL includes 2 configuration files in the *./etc* directory, *userdata_user.conf* and *userdata_main.conf*. One of these files is copied to *userdata.conf* during installation.This parameter is set internally.

Monitoring projects and jobs using a Web browser

You can use a Web browser to monitor the state of all grid nodes, including the ETL Development grid engine that is a part of the grid architecture. In addition to monitoring projects and jobs that are launched from ETL Development, you can:

- Monitor the state of a remote job.
- Suspend and resume a remote job and project.
- View the remote log file of an ETL Server.
- View alert history
- View the list of scheduled tasks

Before you start monitoring:

- Start the ETL Server, if it is not already running.
- Verify that you have Internet Explorer (IE) 6.0 or later installed on your machine.

❖ **Monitoring projects and jobs**

1 Open the Web browser.

2 Enter:

`http://<hostname>:<port_number>`

where <*hostname*> is the network name of the machine on which the ETL Server is running, and <*port_number*> is the port on which the node was started. The default port number is 5124.

On the monitoring page you see:

- Node Summary – displays detailed information about the currently running server, such as the host name and operating system of the machine the server is running on, the number of jobs running on the server, the amount of CPU, memory, and disk space utilized by the server, PID and account information, and the product name and version number. This tab also includes a list of active and recent jobs.
- Active Jobs – displays a detailed list of the running jobs.
- Job History – provides a list of all the jobs executed since the previous day.
- Log History – provides the system log history.
- Node Overview– provides a list of all the running servers.
- Alert History – provides the system alert history.

❖ **Viewing active jobs**

- Click the Active Jobs tab. You see all the running jobs along with details such as name, status, number of projects within the job, start and end time, and the number of records processed.

❖ **Suspending an active job**

- In the Active Jobs tab or the Node Summary tab, click the Suspend icon next to the job you want to suspend. The status of the job changes to Suspended. All projects in the job are also suspended.

❖ **Resuming a job**

1 In the Active Jobs tab or the Node Summary tab, select a job to view its history. Alternatively, you can click the Job History tab.

3 Click the Resume icon next to the suspended job.

4 The status of the job changes to Running and all projects in the job are also resumed.

❖ **Canceling an active job**

- In the Summary tab, click the Cancel icon next to the job you want to cancel. The job is removed from the list.

❖ **Suspending an active project**

- In the Active Jobs tab, click the Suspend icon next to the project you want to suspend. The status of the project changes to Suspended.

❖ **Resuming a project**

1 In the Active Jobs tab, select the job that has a suspended project. Alternatively, you can click the Job History tab.

2 Click the Resume icon next to the suspended project. The projects status changes to Running and the project is resumed.

❖ **Canceling a project**

1 In the Active Jobs tab, select a job from the list of active jobs.

2 Click the Cancel icon next to the project you want to cancel.

The project status changes to Cancelled and the project is stopped.

❖ **Viewing all running servers**

- Select the Node Overview tab. A list of all the running servers appears along with details such as status, number of jobs, amount of CPU, memory, and disk space utilized by the server.

❖ **Shutting down a server**

- In the Node Overview tab, click the Shutdown icon next to the server. The server is removed from the list.

❖ **Viewing log history**

- Select the Log History tab. A table appears, displaying system log details such as timestamp, type of error, and the error message.

  If required, click Download to download the log file on to your machine.

❖ **Viewing the list of scheduled tasks**

To view the list of scheduled tasks:

Note Tasks scheduled on Windows Scheduler are not displayed.

---

1 Select a grid engine to view the list of tasks scheduled on it. All running engines are displayed on the left side of the page.

2 Select the Schedule Task List tab. A list of scheduled tasks available on the selected engine is displayed.

3 Click the Terminate icon next to the task to stop it. Similarly, click the Start icon next to a terminated task to execute it.

❖ **Viewing alert history**

- Select the Alert History tab. A table appears displaying alert details.

  If required, click Download to download the alert log file on to your machine.

Troubleshooting Sybase ETL Server

If you encounter issues with Sybase ETL Server, you can contact Sybase Technical Support for assistance. Before you contact them:

1 Review the error text.

2 Review the log file.

3 Run it again with system trace enabled.

4 Verify the version and revision number as well as your machine ID using:

```
GridNode --version
```

Output:

```
GridNode 4.9.0.27537
```

5 Verify the available licenses using:

```
GridNode --licenses
```

Output:

```
GridNode (4.9.0.27537)
Grid Node
-------------------------------------------------
Product ID: SybaseETLServer
Machine ID: 9TuA+igB6298Hys=
SYSAM ID  : 001111eb57f9 DISK_SERIAL_NUM=189e22a0
-------------------------------------------------
Install Date: Tuesday July 03 13:53:47 2009
-------------------------------------------------
```

```
File: sy_etl_server.license
Product: Sybase ETL Server (SybaseETLServer)
Version: 4.9
License: ETL Components 4.9(ETL_SERVER)
Status: Valid
```

APPENDIX A

Function Reference

Aggregation

Function	Description
uAvg	Returns the average value over all input values
uMax	Returns the maximum from a list of values
uMin	Returns the minimum from a list of values

uAvg

Description: Returns the average value over all input values

Syntax: real uAvg(value, ...)

Parameters:

number value
A list of numeric arguments

Examples:

```
uAvg(1,2,3,4,5) // returns 3
```

uMax

Description: Returns the maximum from a list of values

Syntax: uMax(value,...)

Parameters:

number value
A list of numeric arguments

Examples:

```
uMax(1, 6, 4, -6) // returns 6

uMax("b", "A", "a") // returns "b"

uMax("2004-05_02", "2006-12-12", "1999-05-30") //
returns "2006-12-12"
```

uMin

Description: Returns the minimum from a list of values

Syntax: uMin(value, ...|)

Parameters:

number value
A list of numeric arguments

Examples:

```
uMin(1, 6, 4, -6) // returns -6

uMin("b", "A", "a") // returns "A"

uMin("2004-05-02", "2006-12-12", "1999-05-30")
//returns "1999-05-30"
```

Bit operations

Function	Description
uBitAnd	Bitwise AND operation
uBitOr	Bitwise OR operation

uBitAnd

Description: Bitwise AND operation

Syntax: number uBitAnd(value, ...)

Parameters: *number value*
A list of numeric arguments

Examples:
```
uBitAnd(10, 3) // returns "2"
```

uBitOr

Description: Bitwise OR operation

Syntax: number uBitOr(value, ...)

Parameters: *number value*
A list of numeric arguments

Examples:
```
uBitOr(10, 3) // returns "11"
```

Boolean

Function	Description
uIsAscending	Returns 1 if every parameter is equal to or greater than its predecessor
uIsBoolean	Returns 1 if the parameter is one of 1, true, or yes

Function	Description
uIsDate	Returns 1 if the parameter can be interpreted as a date
uIsDescending	Returns 1 if every parameter is equal to or lower than its predecessor
uIsEmpty	Returns 1 if the parameter is empty or null
uIsInteger	Returns 1 if the parameter can be interpreted as an integer value
uIsFloat	Returns 1 if the parameter can be interpreted as a floating point value
uIsNull	Returns 1 if the parameter is null
uIsNumber	Returns 1 if the parameter can be interpreted as a number

uIsAscending

Description — Returns 1 if every parameter is equal to or greater than its predecessor

Syntax — number uIsAscending(params, ...)

Parameters

params
A list of expressions or values of any datatype.

Examples — Check multiple values for an ascending order:

```
uIsAscending("A", "B", "C") // returns 1

uIsAscending("A", "A", "C") // returns 1

uIsAscending("A", "C", "B") // returns 0


uIsAscending("1", "2", "3") // returns 1

uIsAscending("3", "2", "2") // returns 0

uIsAscending("2004-03-03", "2004-03-05", "2004-03-07")
// returns 1

uIsAscending("2004-03-03", "2004-03-07", "2004-03-05")
//returns 0
```

uIsBoolean

Description

Returns 1 if the parameter is:

- 1, true, or yes
- 0, no, or false

Syntax

number uIsBoolean(param)

Parameters

param
An expressions or value of any data type.

Examples

Check for Boolean value:

```
uIsBoolean("1")     // returns 1
uIsBoolean("yes")   // returns 1
uIsBoolean("true")  // returns 1
uIsBoolean("-1")   // returns 0
uIsBoolean("0")  // returns 1
```

uIsDate

Description

Returns 1 if the parameter can be interpreted as a date. If the second parameter is omitted, the function tries to apply one these formats:

- `y-M-D H:N:S.s`
- `y-M-D H:N:S`
- `y-M-D`
- `H:N:S`

Note For details about the format string, see the uConvertDate function.

Syntax

number uIsDate(datestring [, format])

Parameters

string datestring
The string to be checked

string format (optional)
The format of the input date

Examples

```
uIsDate("2004-02-29") // returns 1

uIsDate("2003-02-29") // returns 0, since 2003 was not
a leap year
```

uIsDescending

Description: Returns 1 if every parameter is equal to or lower than its predecessor

Syntax: number uIsDescending(params, ...)

Parameters:

params
A list of expressions or values of any data type

Examples: Check multiple values for a descending order:

```
uIsDescending("C", "B", "A") // returns 1

uIsDescending("C", "C", "A") // returns 1

uIsDescending("A", "C", "B") // returns 0

uIsDescending("3", "2", "1") // returns 1

uIsDescending("3", "2", "3") // returns 0

uIsDescending("2004-03-20", "2004-03-15", "2004-03-
07") // returns 1

uIsDescending("2004-03-20", "2004-03-07", "2004-03-
15") // returns 0
```

uIsEmpty

Description: Returns 1 if the parameter is empty or null

Syntax: number uIsEmpty(param)

Parameters:

param
An expression or value to investigate

Examples

```
uIsEmpty("1")           // returns 0

uIsEmpty(null)          // returns 1

uIsEmpty("")            // returns 1
```

uIsInteger

Description: Returns 1 if the parameter can be interpreted as an integer value

Syntax: number uIsInteger(param)

Parameters: *param*
An expression or value to investigate

Examples:

```
uIsInteger ("1")    // returns 1
uIsInteger ("2.34") // returns 0
uIsInteger ("ABC")  // returns 0
```

uIsFloat

Description: Returns 1 if the parameter can be interpreted as a floating point value

Syntax: number uIsFloat(param)

Parameters: *param*
An expression or value to investigate

Examples:

```
uIsFloat("1")    // returns 1
uIsFloat("2.34") // returns 1
uIsFloat("ABC")  // returns 0
```

uIsNull

Description: Returns 1 if the parameter is null

Syntax: number uIsNull(param)

Parameters: *param*
An expression or value to investigate

Examples:

```
uIsNull("1")  // returns 0
uIsNull(null) // returns 1
```

uIsNumber

Description: Returns 1 if the parameter can be interpreted as a number

Syntax: number uIsNumber(param)

Parameters: *param*
An expression or value to investigate

Examples: Check for a numeric value:

```
uIsNumber("1")    // returns 1
uIsNumber("2.34") // returns 1
uIsNumber("ABC")  // returns 0
```

Conversion

Function	Description
uBase64Decode	Decodes a string from a Base64 representation
uBase64Encode	Encodes a string into a Base64 representation
uConvertDate	Converts a date string into the default or a custom date format
uFromHex	Converts a hexadecimal value into an integer value
uToHex	Converts an integer value into a hexadecimal digit
uHexDecode	Composes a string from hexadecimal values
uHexEncode	Encodes the characters of a string into hexadecimal notation
uToUnicode	Converts a string into its Unicode representation
uURIDecode	Decodes a string, replacing escape sequences with their original values
uURIEncode	Replaces certain characters in a URI with escape sequences

uBase64Decode

Description: Decodes a string from a Base64 representation

Syntax: string uBase64Decode(input)

Parameters

string input
The string to decode

Examples

```
uBase64Decode("QSBzZWNyZXQ=") // returns "A secret"
```

uBase64Encode

Description

Encodes a string into a Base64 representation

Syntax

string uBase64Encode(input)

Parameters

string input
The string to encode.

Examples

```
uBase64Encode("A secret") // returns "QSBzZWNyZXQ="
```

uConvertDate

Description

Converts a date string into the default or a custom date format.

The function handles dates from 1582 to present day. If the date cannot be converted, the result string is empty.

Syntax

string uConvertDate(datestring, inputformat [, outputformat])

Parameters

string datestring
The date string to convert

string inputformat
The date/time format of the input string

string outputformat (optional)
The desired output format. If omitted, the default format is `y-M-D H:N:S`.

Examples

Convert date strings into a different formats:

```
uConvertDate("2005-06-27 00:00:00","y-M-D H:N:S","D
mY") // returns "27 JUN 05"
```

```
uConvertDate("27 JUN 05", "D m Y") // returns "2005-06-
27 00:00:00"
```

Sybase recommends you to provide input data for all the fields required in the output data. Having fields in the output data for which no input data has been provided, can lead to unexpected results. For example, if you require hours, minutes, and seconds fields in the output data, your input data should be something similar to this:

```
uConvertDate("27 JUN 05", "D m Y", "y-m-D 00:00:00") // returns "2005-06-27
00:00:00"
```

Usage

The function uConvertDate converts a date string into a different format using a source and a destination format string. The first parameter is the date string to be converted. The second parameter is a format string, specifying the date format of the input date (see list below). The outputformat parameter is optional; If omitted the date is converted using the format `y-M-D H:N:S`. The function handles dates from 1582 to present day. If the date cannot be converted, the result string is empty.

Identifier	Description
`Y`	Year 2-digits (06)
`y`	Year 4-digits (2006)
`C`	Century (20)
`M`	Month (03)
`m`	Month (JUN)
`D`	Day (12)
`H`	Hour (00 – 23)
`h`	Hour (01 – 12)
`N`	Minutes
`n`	Month (June)
`S`	Seconds
`s`	Hundredth of seconds
`t`	Thousandth of seconds
`A`	AM/PM
`d`	Day of month (05)
`E`	Day of year (001 – 366)
`G`	Week of year (01 – 52)
`F`	Week of month (1 – 6)

uFromHex

Description: Converts a hexadecimal value into an integer value

Syntax: integer uFromHex(input)

Parameters: *string input*
The string to convert:

Examples:

```
uFromHex("A3F") // returns 2623
uFromHex("B")   // returns 11
```

uToHex

Description: Converts an integer value into a hexadecimal value

Syntax: string uToHex(input)

Parameters: *integer input*
The integer value to convert:

Examples:

```
uToHex(45) // returns "2D"
```

uHexDecode

Description: Composes a string from hexadecimal values

Syntax: string uHexDecode(input)

Parameters: *string input*
The hexadecimal string containing the hexadecimal values

Examples: Convert a hexadecimal value to a string:

```
uHexDecode("313730") // returns "170"
uHexDecode(313730)   // returns "170"
```

uHexEncode

Description: Encodes the characters of a string into hexadecimal notation

Syntax: string uHexEncode(input)

Parameters:

string input
The string to encode

Examples: Convert a string to hexadecimal values:

```
uHexEncode("170") // returns "313730"
uHexEncode(170)   // returns "313730"
```

uToUnicode

Description: Converts a string into its Unicode representation

Syntax: string uToUnicode(input)

Parameters:

string input
The input string

uURIDecode

Description: Decodes a string, replacing escape sequences with their original values

Syntax: string uURIDecode(uri)

Parameters:

string uri
The URI to decode

Examples:

```
uURIDecode("www.myServer.com/filename%20with%20spaces.
txt") // returns
"www.myServer.com/filename with spaces.txt"
```

uURIEncode

Description: Replaces certain characters in a URI with escape sequences

Syntax string uURIEncode(uri)

Parameters *string uri*
The URI to encode

Examples
```
uURIEncode("www.myServer.com/filename with
spaces.txt")// returns
"www.myServer.com/filename%20with%20spaces.txt"
```

Date and Time

Most Date and Time functions are derived from the uFormatDate function. The only difference is that the other Date and Time functions return only a special format or part of the date and they do not have the first format parameter. Therefore, uDate() is equivalent to uFormatDate("%Y-%m-%d").

See also
- “Time Strings” on page 243
- “Modifiers” on page 244
- “Date and time calculations” on page 245
- “Known limitations” on page 246
- “Date and time function list” on page 246

Time Strings

Description A time string can be in any of these formats:

1 *YYYY-MM-DD*

2 *YYYY-MM-DD HH:MM*

3 *YYYY-MM-DD HH:MM:SS*

4 *YYYY-MM-DD HH:MM:SS.SSS*

5 *HH:MM*

6 *HH:MM:SS*

7 *HH:MM:SS.SSS*

8 now

9 *DDDD.DDDD*

Note
Formats 5 – 7 that specify only a time assume a date of 2000-01-01. Format 8 is converted into the current date and time, using Universal Coordinated Time (UTC). Format 9 is the Julian day number expressed as a floating point value.

Examples

Getting the current time If no date is given, the time string now is assumed and the date is set to the current date and time.

```
uDate() // returns something like "2006-03-01"
uDate() is equivalent to uDate("now")
```

Getting a special date

```
uDate("2004-01-04 14:26:33")
// returns the date part "2004-01-04"
```

Modifiers

Description

The time string can be followed by zero or modifiers that alter the date or alter the interpretation of the date. The available modifiers are:

1 *NNN* days

2 *NNN* hours

3 *NNN* minutes

4 *NNN.NNNN* seconds

5 *NNN* months

6 *NNN* years

7 start of month

8 start of year

9 start of day

10 weekday *N*

11 unixepoch

12 localtime

13 utc

Examples

Modifiers 1 – 6 simply add the specified amount of time to the date specified by the preceding time string.

The "start of" modifiers (7 – 9) shift the date backward to the beginning of the current month, year, or day.

The "weekday" (10) modifier advances the date forward to the next date where the weekday number is *N*: Sunday is 0, Monday is 1, and so on.

The unixepoch modifier (11) works only if it immediately follows a time string in the *DDDD.DDDDD* format. This modifier causes the *DDDD.DDDDD* to be interpreted not as a Julian day number as it normally would be, but as the number of seconds since 1970. This modifier allows UNIX-based times to be easily converted to Julian day numbers.

The localtime modifier (12) adjusts the previous time string so that it displays the correct local time. utc undoes this.

Date and time calculations

Description

These examples show you some typical date and time calculations.

Examples

Compute the current date:

```
uDate('now')
```

Compute the last day of the current month:

```
uDate('now','start of month','+1 month','-1 day')
```

Compute the date and time given a UNIX timestamp 1092941466:

```
uDatetime(1092941466, 'unixepoch')
```

Compute the date and time given a UNIX timestamp 1092941466, and compensate for your local time zone:

```
uDatetime(1092941466, 'unixepoch', 'localtime')
```

Compute the current UNIX timestamp:

```
uFormatDate ('%s','now')
```

Compute the number of seconds between two dates:

```
uJuliandate('now')*86400 - uJuliandate ('2004-01-01 02:34:56')*86400
```

Compute the date of the first Tuesday in October (January + 9) for the current year:

```
uDate('now','start of year','+9 months','weekday 2')
```

Known limitations

Description

The computation of local time varies by locale. The standard C library function localtime() is used to assist in the calculation of local time. Also, the localtime() C function normally only works for years between 1970 and 2037. For dates outside this range, we attempt to map the year into an equivalent year within this range, do the calculation, then map the year back.

- Date computations do not give correct results for dates before Julian day number 0 (-4713-11-24 12:00:00).
- All internal computations assume the Gregorian calendar system.

Date and time function list

Description

This table lists all the date and time functions.

Function	Description
uDate	Returns a year, month, and day from a date in the format *YYYY-MM-DD*
uDateTime	Returns a year, month, and day from a date in the format *YYYY-MM-DD HH.MM.SS*
uDay	Returns the day number of the date specified
uDayOfYear	Returns the number of days since the beginning of the year
uHour	Returns the hour of the date specified
uQuarter	Returns the quarter of the year
uIsoWeek	Returns the week number defined by ISO 8601
uJuliandate	Returns the number of days since noon in Greenwich on November 24, 4714 B.C, in the format *DDDD.DDDD*
uMinute	Returns the minute of the date specified
uMonth	Returns the month of the name specified
uMonthName	Returns the name of month of the date specified, in the current locale language
uMonthNameShort	Returns the short form of the name of month of the date specified, in the current locale language
uSeconds	Returns the second of the date specified
uTime	Returns the time part from a date in, the format *HH.MM.SS*
uTimeDiffMs	Returns the difference between two dates, in milliseconds
uWeek	Returns the week of the date specified
uWeekday	Returns the weekday of the date specified
uWeekdayName	Returns the weekday name of the date specified, in the current locale language
uWeekdayNameShort	Returns the short form of the weekday name of the date specified in the current locale language
uYear	Returns the year of the date specified

Note Refer to “Date and Time” on page 243 for detailed information about the possible modifier arguments.

uDate

Description — Returns a year, month, and day from a date in the format *YYYY-MM-DD*

Syntax — string uDate([modifiers])

Parameters — *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples — Get the date part out of a timestamp:

```
uDate("now") // returns current date in the form "YYYY-MM-DD".

uDate("now", "start of year", "9 months", "weekday 2") // returns the date of the first Tuesday in October this year.
```

uDateTime

Description — Returns a year, month, and day from a date in the format *YYYY-MM-DD HH.MM.SS*

Syntax — string uDateTime([modifiers])

Parameters — *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples — Get the datetime part from a timestamp:

```
uDateTime("now") // returns current date in the form "YYYY-MM-DD HH:MM:SS"

uDateTime("now", "start of month", "1 months", "-1 day") // returns the date of the last day in this month
```

uDay

Description — Returns the day number of the date specified

Syntax — string uDay([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples Get the day number out of a timestamp:

```
uDay("now") // returns current day number

uDay("1969-03-13 10:22:23.231") // returns "13"
```

uDayOfYear

Description Returns the number of days since the beginning of the year

Syntax string uDayOfYear([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples Get the day number out of a timestamp:

```
uDayOfYear("now") // returns how many days have already
passed this year

uDayOfYear("1969-03-13 10:22:23.231") // returns "72"
```

uHour

Description Returns the hour of the date specified

Syntax string uHour([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uHour("now") // returns current hour

uHour("1969-03-13 10:22:23.231") // returns "10"
```

uQuarter

Description — Returns the quarter of the year

Syntax — string uQuarter([modifiers])

Parameters — *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uQuarter ("now") // returns current quarter
uQuarter ("2005-03-13 10:22:23.231") // returns "1"
```

uIsoWeek

Description — Returns the week number defined by *ISO 8601*

The first week of a year is number 01, which is defined as being the week that contains the first Thursday of the calendar year, which implies that it is also:

- The first week that is mostly within the calendar year
- The week containing January 4th
- The week starting with the Monday nearest to January 1st

The last week of a year, number 52 or 53, therefore is:

- The week that contains the last Thursday of the calendar year
- The last week that is mostly within the calendar year
- The week containing December 28th
- The week ending with the Sunday nearest to December 31st

Syntax — number uIsoWeek([modifiers])

Parameters — *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uIsoWeek("now") // returns current week number
```

uJuliandate

Description
Returns the number of days since noon in Greenwich on November 24, 4714 B.C, in the format *DDDD.DDDD*. For date and time calculation, the juliandate function is the best choice.

Syntax
string uJuliandate([modifiers])

Parameters
string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples
Convert a date into a numerical value for calculation:

```
uJuliandate("now") // returns current date in the form
"DDDD.DDDD"
```

Compute the number of seconds between two dates:

```
uJuliandate('now')*86400 - uJuliandate('2004-01-01
02:34:56')*86400
```

Compute the date and time given a UNIX timestamp 1092941466, and compensate for your local time zone:

```
uJuliandate(1092941466, 'unixepoch', 'localtime')
```

uMinute

Description
Returns the minute of the date specified

Syntax
string uMinute([modifiers])

Parameters
string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uMinute("now") // returns current minute
```

```
uMinute("1969-03-13 10:22:23.231") //returns "22"
```

uMonth

Description
Returns the month of the date specified

Syntax string uMonth([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples
```
uMonth("now") // returns current month

uMonth("1969-03-13 10:22:23.231") // returns "03"
```

uMonthName

Description Returns the name of month of the date specified, in the current locale language

Syntax string uMonthName([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples Get the name of month from a date:

```
uMonthName("now") // returns current name of month
```

Set the locale to English:

```
uSetLocale("English")
uMonthName("1969-03-13 10:22:23.231") // returns
"March"
```

Set the locale to German:

```
uSetLocale("German")
uMonthName("1969-03-13 10:22:23.231") // returns "März"
```

uMonthNameShort

Description Returns the short form of the name of month of the date specified, in the current locale language

Syntax string uMonthNameShort([modifiers])

Parameters *string modifiers (optional)*
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

Get the name of month from a date:

```
uMonthNameShort("now") // returns current name of
month.
```

Set the locale to English:

```
uSetLocale("English")
uMonthNameShort("1969-03-13 10:22:23.231") // returns
"Mar"
```

Set the locale to German:

```
uSetLocale("German")
uMonthNameShort("1969-03-13 10:22:23.231") // returns
"Mär"
```

uSeconds

Description

Returns the second of the date specified.

Syntax

string uSeconds([modifiers])

Parameters

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uSeconds("now") // returns current second
```

```
uSeconds("1969-03-13 10:22:23.231") // returns "23"
```

uTime

Description

Returns the time part from a date, in the format *HH.MM.SS*.

Syntax

string uTime([modifiers])

Parameters

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

Get the time part from a timestamp:

```
uTime() // returns current UTC time
```

```
uTime("now","localtime") // returns current local time
```

uTimeDiffMs

Description: Returns the difference between two dates, in milliseconds

Syntax: string uTimeDiffMs(date1, date2)

Parameters:

string date1
The older date

string date2
The more recent date

Examples:

```
uTimeDiffMs ("18:34:20", "18:34:21")     // returns 1000
uTimeDiffMs ("18:34:20", "18:34:21.200") // returns
1200
```

uWeek

Description: Returns the week of the date specified

Syntax: string uWeek([modifiers])

Parameters:

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples:

```
uWeek("now") // returns current week
uWeek("1969-03-13 10:22:23.231") // returns "10"
```

uWeekday

Description: Returns the weekday number of the date specified

Syntax: string uWeekday([modifiers])

Parameters

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uWeekday("now") // returns current weekday number

uWeekday("1969-03-13 10:22:23.231") // returns "4" for
Thursday
```

uWeekdayName

Description

Returns the weekday name of the date specified, in the current locale language

Syntax

string uWeekdayName([modifiers]);

Parameters

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uWeekdayName("now") // returns current weekday name
```

Set the locale to English:

```
uSetLocale("English")
uWeekdayName("1969-03-13 10:22:23.231") // returns
"Thursday"
```

Set the locale to German:

```
uSetLocale("German")
uWeekdayName("1969-03-13 10:22:23.231") // returns
"Donnerstag"
```

uWeekdayNameShort

Description
Returns the short form of the weekday name of the date specified, in the current locale language

Syntax
string uWeekdayNameShort([modifiers])

Parameters
string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uWeekdayNameShort("now") // returns current weekday
name
```

Set the locale to English:

```
uSetLocale("English")
uWeekdayNameShort("1969-03-13 10:22:23.231") //returns
"Thu"
```

Set the locale to German:

```
uSetLocale("German")
uWeekdayNameShort("1969-03-13 10:22:23.231") //returns
"Don"
```

uYear

Description
Returns the year of the date specified

Syntax
string uYear([modifiers])

Parameters
string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples

```
uYear("now") // returns current year
```

```
uYear("1969-03-13 10:22:23.231") // returns "1969"
```

Error handling

Function	Description
uError	Writes an error text into a log and signals an error
uErrortext	Returns the last error message
uWarning	Writes a warning message into a log
uInfo	Writes an informal message into a log
uTrace	Writes a trace message into a log
uTracelevel	Sets the detail level of trace messages in the a log

uError

Description — Writes an error text into a log and signal an error

Syntax — string uError(errortext)

Parameters — *string errortext*
Text to write to log file

Examples — Signal an error:

```
uError("'PP' is no valid country key.")
```

uErrortext

Description — Returns the last error message

Syntax — string uErrortext()

Examples

```
uErrortext() // returns last error text
```

uInfo

Description — Writes an informal message into a log

Syntax — string uInfo(infotext)

Parameters — *string infotext*
Text to write to log file

Examples — Log an informal message:

```
uInfo("21445 records selected.")
```

uWarning

Description — Writes a warning message into a log

Syntax — string uWarning(warningtext)

Parameters — *string warningtext*
Text to write to log file

Examples — Log a warning message:

```
uWarning("The attribute for the customer name is null.")
```

uTrace

Description — Writes a trace message into a log.

You must manually set the trace level to at least 1before invoking the uTrace() function. To set the trace level to 1, do one of these:

- Invoke uTracelevel(1) before invoking the uTrace() function.
- If you are using ETL Development, set the trace level to 1 in the *Default.ini* file located in the *etc* directory of the installation folder. Restart ETL Development.
- If you are using ETL Server, start the server with the "--tracelevel 1" option.

Syntax — string uTrace(tracetext);

Parameters

string tracetext
Text to write to log file

Examples

```
uTrace("CUSTOMER_NAME = " + CUSTOMER_NAME)
```

uTracelevel

Description

Sets the detail level of trace messages in the log. The range of tracelevel is from 0 (no trace) to 5 (very verbose).

Syntax

uTracelevel(tracelevel)

Note Verbose message tracing may dramatically reduce performance.

Parameters

integer tracelevel
Specifies the verbosity of trace messages. (0 = off, 5 = very verbose)

Examples

```
uTracelevel(5) // sets the tracelevel to 'very verbose'
```

Files

Function	Description
uFileInfo	Returns information about a file
uFileRead	Reads data from a file
uFileWrite	Writes data to a file

uFileInfo

Description

Returns information about a file. When infotype is set to `EXISTS`, the function returns the entire path to the file, if it exists, or an empty string if it does not. If infotype set to `SIZE`, the size of the file is returned. If the file does not exist, an empty string is returned.

Note Use double backslashes in JavaScript environments, because the backslash is used as escape sequence.

Syntax

string uFileInfo(file [, infotype])

Parameters

string file
The file to investigate

string infotype (optional)
The kind of information to get. Default is `EXISTS`.

Examples

Get file information:

```
uFileInfo("C:\\windows\\notepad.exe") // returns
C:\windows\notepad.exe

uFileInfo("C:\\windows\\notepad.exe","SIZE") //
returns 68608
```

uFileRead

Description

Reads data from a file

Syntax

string uFileRead(URL [, bytes] [, offset] [, encoding])

Parameters

string URL
URL specifying the source to read

integer bytes (optional)
Number of bytes to read. Default is 0, which means the entire file.

integer offset(optional)
Number of bytes to skip from the beginning of the file. Default is 0.

string encoding(optional)
The encoding of the data source. Default encoding is ISO8859-1.

Examples

Access local files:

```
uFileRead("c:\\myFile.txt")

uFileRead("/home/testuser/myfile.txt")

uFileRead("file:///c:/ myFile.txt")
```

Read files from a Windows share:

```
uFileRead("\\\\fileserver\\freeShare\\testfile.txt")
```

Read the content of a file via HTTP and HTTPS:

```
uFileRead("http://http://www.google.com/search?hl=en&q
pizza&btnG=Google+Search")

uFileRead("https://http://www.google.com/search?hl=en&
q=pizza&btnG=Google+Search")
```

Read the content of a file via FTP:

```
uFileRead("ftp://myUser:myPasswd@myServer/data/myFile.
txt")
```

uFileWrite

Description — Writes data to a file. If no URL is given, the data is written to a file *write.log* in the Sybase ETL log directory.

Syntax — string uFileWrite(data [, URL] [, append] [, encoding])

Parameters

string data
The data to be written

string URL (optional)
URL for file access and location

number append (optional)
Flag (0/1) indicating if the data should be appended or not

string encoding (optional)
The encoding of the target file

Examples — Write data to a file via Common Internet File System (CIFS):

```
uFileWrite("hello",
"//myServer/myShare/data/test.txt")
```

Formatting

Function	Description
uFormatDate	Returns a user-defined string with date information.

uFormatDate

Description — Returns a user-defined string with date information.

Syntax — number uFormatDate(format, modifiers, ...)

Parameters

string format
A format specification for the return string

string modifiers (optional)
List of strings specifying a date or date calculation. Default is the now modifier.

Examples — Create a string from a date:

```
uFormatDate("Today is %A the %d of %B in %Y", "now")
//returns something like "Today is Thursday the 10 of
February in 2005"
```

Usage — Special escape sequences in the user-defined format string are replaced by the referring date part.

Escape sequence %A	Returns Weekday name
%a	Weekday name short
%B	Month name
%b	Month name short
%d	Day of month
%f	Fractional seconds SS.SSS
%H	Hour 00 – 24
%j	Day of year 000 – 366
%J	Julian day number
%m	Month
%M	Minute

Escape sequence %A	Returns Weekday name
%s	Seconds since 1970 – 01 – 01
%S	Seconds 00 – 59
%w	Day of week 0 – 6, 0=Sunday
%W	Week of year
%Y	Year 0000 – 9999
%%	%

Fuzzy Search

Function	Description
uGlob	Compares case-sensitive values that are similar, using the UNIX file globbing syntax for its wildcards
uLike	Compares values case insensitive
uMatches	Returns true if a given string matches a regular expression

uGlob

Description — Compares case-sensitive values that are similar, using the UNIX file globbing syntax for its wildcards.

Syntax — bool uGlob(pattern, text)

Parameters

string pattern
A string describing a match pattern

string text
A string to investigate

Examples — Compare values using UNIX file globbing syntax:

```
uGlob("Mr. *", "Mr. Smith")   // returns 1, indicating
a match

uGlob("Mr. *", "Mrs. Clarke") // returns 0
```

Globbing rules:

"*" – matches any sequence of zero or more characters.

"?" – matches exactly one character.

[^...] – matches one character not in the enclosed list.

[...] – matches one character from the enclosed list of characters.

With [...] and [^...] matching, a closing square bracket ( ]) can be included in the list by making it the first character after an opening square bracket ([ ) or a caret (^). Specify a range of characters using a hyphen (-):

- "[a-z]" matches any single lowercase letter. To match a hyphen (-), make it the last character in the list.
- To match an asterisk (*) or a question mark (?), place them in square brackets ([]).

  For example: abc[*]xyz, matches the literal value "abc*xyz" .

uLike

Description

Compares values case insensitive.

The uLike function performs a pattern-matching comparison. The first parameter contains the pattern, the second parameter contains the string to match against the pattern. A percent symbol (%) in the pattern matches any sequence of zero or more characters in the string. An underscore (_) in the pattern matches any single character in the string. Any other character matches itself or its lowercase or uppercase equivalent.

Note Currently, uLike only interpret only uppercase and lowercase for 7-bit Latin characters, which means uLike is case-sensitive for 8-bit ISO8859 characters or UTF-8 characters. For example:
`uLike('a' ,'A')` returns 1.
`uLike('æ' ,'Æ')` returns 0.

Syntax

number uLike(pattern, text)

Parameters

string pattern
A string describing a match pattern

string text
A string to investigate

Examples

Compare values using pattern matching:

```
uLike("% happy %", "A happy man.") // returns 1
uLike("% happy %", "A sad man.")   // returns 0
```

uMatches

Description

Returns true if a given string matches a regular expression.

Syntax

number uMatches(text, regexpr)

Parameters

string text
Text to investigate

string regexpr
Regular expression specification

Examples

Check if a string could be interpreted as a floating point number:

```
uMatches("abc","[-+]?[0-9]*\\.?[0-9]*")  // returns 0
uMatches("1.23","[-+]?[0-9]*\\.?[0-9]*")  // returns 1
```

Lookup

Function	Description
uChoice	Returns the value of a given parameter specified by an index
uFirstDifferent	Returns the first parameter value that differs from the first parameter
uFirstNotNull	Returns the first non-null parameter
uElements	Returns the number of elements in a delimited string
uToken	Returns the Nth element from a delimited string

uChoice

Description — Returns the value of a given parameter specified by an index. The index value is zero-based, so an index of zero returns the second parameter.

Syntax — string uChoice(index, values, ...)

Parameters

integer index
Zero based index number referencing the return value.

string values
List of values

Examples — IF construct:

```
uChoice(0, "A", "B") // returns "A"

uChoice(1, "A", "B") // returns "B"
```

CASE construct:

```
uChoice(2, "n.a.", "Jan", "Feb", "Mar") //returns "Feb"
```

Simulate a lookup function, where you want to replace a color ID with a corresponding color name:

```
uChoice(IN.Color, "n.a.", "Red", "Blue", "Green")
```

uFirstDifferent

Description — Returns the first parameter value that differs from the first parameter

Syntax — string uFirstDifferent(params, ...)

Parameters

params
A list of expressions or values of any data type

Examples

```
uFirstDifferent("2004-05-01", "2004-05-01", "2005-01-
04", "2005-11-24",) //returns "2005-01-04"
```

uFirstNotNull

Description — Returns the first non-null parameter.

Syntax — string uFirstNotNull(params, ...)

Parameters

params
A list of expressions or values of any data type

Examples

```
uFirstNotNull(null, null , "A", "B") // returns "A"
```

uElements

Description

Returns the number of elements in a delimited string. If the second parameter is omitted, a space (ASCII 32) is taken as a delimiter.

Syntax

integer uElements(text [, delimiter])

Parameters

string text
A string to investigate

string delimiter (optional)
The delimiter to be used. Default delimiter is a space character.

Examples

Count tokens in a delimited string:

```
uElements("James T. Kirk") // returns 3
```

uToken

Description

Returns the Nth element from a delimited string. The second parameter specifies the token number. The index starts at 1. If the third parameter is omitted, a space (ASCII 32) is used as the delimiter.

Syntax

string uToken(text, index [, delimiter])

Parameters

string text
A string to investigate

Integer index
Number of tokens to be returned

string delimiter (optional)
The delimiter to be used. Default delimiter is a space character.

Examples

```
uToken("James T. Kirk", 1) // returns "James"
uToken("James T. Kirk", 2) // returns "T."
```

Miscellaneous

Function	Description
uCommandLine	Returns the command line string of the current process
uGetEnv	Returns the value of an environment variable
uGuid	Returns a global unique identifier
uMD5	Generates a checksum over a given string
uScriptLoad	Loads and evaluates JavaScript and returns the result
uSetEnv	Sets the value of an environment variable
uSetLocale	Changes the locale date and time settings to a different language
uSleep	Suspends the process for a specified number of milliseconds
uSystemFolder	Returns predefined application and system paths

uCommandLine

Description — Returns the command line string of the current process

Syntax — string uCommandLine()

Examples

```
uCommandLine() // returns
"GridNode.exe --port 5124"
```

Note uCommandLine is not supported on UNIX.

uGetEnv

Description — Returns the value of an environment variable

Syntax — string uGetEnv(variable)

Parameters

string variable
Name of the environment variable to read

Examples

```
uGetEnv("LOAD_MAX_VALUE")
```

uGuid

Description

Returns a global unique identifier in one of these formats:

- *numeric* – digits only
- *base64* – Base64-encoded
- *hex* – hex format without hyphens

Syntax

string uGuid([format])

Parameters

string format (optional)
Format for the GUID value to be returned

Examples

```
uGuid() // returns for example A8A10D9F-963F-4914-
8D6FC8527A50EF2A
```

uMD5

Description

Generates a checksum over a given string with a fixed length of 32 characters

Syntax

string uMD5(text)

Parameters

string text
Text to build a checksum on

Examples

```
uMD5("Austin Powers") // returns
"C679A893E3DA2CC0741AC7F527B1D4EB"
```

uScriptLoad

Description

Loads and evaluates JavaScript and returns the result

Syntax

string uScriptLoad(filelocation)

Parameters

string filelocation
The JavaScript file to load.

Examples

Load an external JavaScript file:

```
uScriptLoad("\\server3\myScripts\basicFunctions.js")
```

uSetEnv

Description

Set the value of an environment variable

Syntax

string uSetEnv(variable, value)

Parameters

string variable
Name of the environment variable to set

string value
Value to set

Examples

```
uSetEnv("LOAD_MAX_VALUE", IN.Date)
```

uSetLocale

Description

Changes the locale date and time settings to a different language

Syntax

string uSetLocale([language] [, country] [, codepage])

Parameters

string language (optional)
Language string to be used (see table in Usage section)

string country (optional)
Country name to be used (see table in Usage section)

string codepage (optional)
Code page number as string

Examples

Retrieve month names in different languages:

```
locale:uSetLocale("english")    // switch to english
uMonthName("2005-03-22") // returns "March"
uSetLocale("german")     // switch to german
uMonthName("2005-03-22") // returns "Marz"
uSetLocale("C")          // switch back to OS default
```

Usage

Language Strings

The language strings in the table below are recognized. Any language not supported by the operating system is not accepted by uSetLocale.

Note The three-letter language-string codes are valid only in Windows NT and Windows 95.

Primary language	Sublanguage	Language string
Chinese	Chinese	"chinese"
Chinese	Chinese (simplified)	"chinese-simplified" or "chs"
Chinese	Chinese (traditional)	"chinese-traditional" or "cht"
Czech	Czech	"csy" or "czech"
Danish	Danish	"dan"or "danish"
Dutch	Dutch (Belgian)	"belgian", "dutch-belgian",or "nlb"
Dutch	Dutch (default)	"dutch" or "nld"
English	English (Australian)	"australian", "ena", or "english-aus"
English	English (Canadian)	"canadian", "enc", or "english-can"
English	English (default)	"english"
English	English (New Zealand)	"english-nz" or "enz"
English	English (UK)	"eng", "english-uk", or "uk"
English	English (USA)	english", "americanenglish", "english-american", "english-us", "english-usa", "enu", "us", or "usa"
Finnish	Finnish	"fin" or "finnish"
French	French (Belgian)	"frb" or "french-belgian"
French	French (Canadian)	"frc" or "frenchcanadian"
French	French (default)	"fra"or "french"
French	French (Swiss)	"french-swiss" or "frs"
German	German (Austrian)	"dea" or "germanaustrian"
German	German (default)	"deu" or "german"

Primary language	Sublanguage	Language string
German	German (Swiss)	"des", "german-swiss", or "swiss"
Greek	Greek	"ell" or "greek"
Hungarian	Hungarian	"hun" or "hungarian"
Icelandic	Icelandic	"icelandic" or "isl"
Italian	Italian (default)	"ita" or "italian"
Italian	Italian (Swiss)	"italian-swiss" or "its"
Japanese	Japanese	"japanese" or "jpn"
Korean	Korean	"kor" or "korean"
Norwegian	Norwegian (Bokmal)	"nor" or "norwegianbokmal"
Norwegian	Norwegian (default)	"norwegian"
Norwegian	Norwegian (Nynorsk)	"non" or "norwegiannynorsk"
Polish	Polish	"plk" or "polish"
Portuguese	Portuguese (Brazil)	"portuguese-brazilian" or "ptb"
Portuguese	Portuguese (default)	"portuguese" or "ptg"
Russian	Russian (default)	"rus" or "russian"
Slovak	Slovak	"sky" or "slovak"
Spanish	Spanish (default)	"esp" or "spanish"
Spanish	Spanish (Mexican)	"esm" or "spanish-mexican"
Spanish	Spanish (Modern)	"esn" or "spanish-modern"
Swedish	Swedish	"sve" or "swedish"
Turkish	Turkish	"trk" or "turkish"

Country or Region Strings

The following is a list of country/regions strings recognized by uSetLocale. Strings for countries/regions that are not supported by the operating system are not accepted by uSetLocale. Three-letter country or region codes are from ISO or IEC (International Organization for Standardization, International Electrotechnical Commission) specification 3166.

Country or region	Country or region string
Australia	"aus" or "australia"
Austria	"austria" or "aut"
Belgium	"bel" or "belgium"

Country or region	Country or region string
Brazil	"bra" or "brazil"
Canada	"can" or "canada"
Czech Republic	"cze" or "czech"
Denmark	"denmark" or "dnk"
Finland	"fin" or "finland"
France	"fra" or "france"
Germany	"deu" or "germany"
Greece	"grc" or "greece"
Hong Kong SAR	"hkg", "hong kong", or "hong-kong"
Hungary	"hun" or "hungary"
Iceland	"iceland" or "isl"
Ireland	"ireland" or "irl"
Italy	"ita" or "italy"
Japan	"japan" or "jpn"
Korea	"kor", "korea"
Mexico	"mex" or "mexico"
Netherlands	"nld", "holland", or "netherlands"
New Zealand	"new zealand", "new-zealand", "nz", or "nzl"
Norway	"nor" or "norway"
People's Republic of China	"china", "chn", "pr china", or "pr-china"
Poland	"pol" or "poland"
Portugal	"prt" or "portugal"
Russia	"rus" or "russia"
Singapore	"sgp" or "singapore"
Slovak Repubic	"svk" or "slovak"
Spain	"esp" or "spain"
Sweden	"swe" or "sweden"
Switzerland	"che" or "switzerland"
Taiwan	"taiwan" or "twn"
Turkey	"tur" or "turkey"
United Kingdom	"britain", "england", "gbr", "great britain", "uk", "united kingdom", or "united-kingdom"
United States of America	"america", "united states", "unitedstates", "us", or "usa"

uSleep

Description: Suspends the process for a specified number of milliseconds

Syntax: string uSleep(msecs)

Parameters: *integer msecs*
Number of milliseconds for which to suspend the process

Examples:

```
uSleep(1000) // suspends the process for one second
```

uSystemFolder

Description: Returns predefined application and system paths

Syntax: string uSystemFolder([foldertype])

Examples:

```
uSystemFolder("APP_LOG") // returns the path to the log
directory
```

Usage: Use uSystemFolder to access a special directory on the file system. You can specify the following folders:

Group	Name	Description
Application	APP_MAIN	The base application path. A typical path is *C:\Program Files\ETL*.
Application	APP_LIB	Shared library directory. A typical path is *application_directory\lib*.
Application	APP_LOG	Shared library directory. A typical path is *application_directory\lib*.
Application	APP_CONFIG	Config file directory. A typical path is *application_directory\etc*.
Application	APP_LICENSE	License directory. A typical path is *application_directory\license*.
Application	APP_SCRIPT	Script directory. A typical path is *application_directory\scripts*.
Application	APP_GRAMMAR	Grammar directory. A typical path is *application_directory\grammar*.
Application	APP_LANGUAGE	Language file directory. A typical path is *application_directory\language*.
Application	APP_DATABASE	Database directory. A typical path is *application_directory\database*.

Group	Name	Description
Application	APP_TEMP	Temporary directory. A typical path is *application_directory\temp*.
Application	APP_DEMODATA	Demodata directory. A typical path is *application_directory\demodata*.
Application	APP_USERDATA	Directory where user specific files are stored. Typical path is *C:\Documents and Settings\username\Application Data\ETL*.
Windows	ALTSTARTUP	The file system directory that corresponds to the user's non-localized Startup program group.
Windows	APPDATA	The file system directory that serves as a common repository for application-specific data. A typical path is *C:\Documents and Settings\username\Application Data*.
Windows	CDBURN_AREA	The file system directory acting as a staging area for files waiting to be written to CD. A typical path is *C:\Documents and Settings\username\Local Settings\Application Data\Microsoft\CD Burning*.
Windows	COMMON_ADMINTOOLS	The file system directory containing administrative tools for all users of the computer
Windows	COMMON_APPDATA	The file system directory containing application data for all users. A typical path is *C:\Documents and Settings\All Users\Application Data*.
Windows	COMMON_DESKTOPDIRECTORY	The file system directory that contains files and folders that appear on the desktop for all users. A typical path is *C:\Documents and Settings\All Users\Desktop*. Valid only for Windows NT systems.
Windows	COMMON_DOCUMENTS	The file system directory that contains documents that are common to all users. A typical paths is *C:\Documents and Settings\All Users\Documents*.
Windows	COMMON_FAVORITES	The file system directory that serves as a common repository for favorite items common to all users. Valid only for Windows NT systems.
Windows	COMMON_MUSIC	The file system directory that serves as a repository for music files common to all users. A typical path is *C:\Documents and Settings\All Users\Documents\My Music*.

Group	Name	Description
Windows	COMMON_PICTURES	The file system directory that serves as a repository for image files common to all users. A typical path is *C:\Documents and Settings\All Users\Documents\My Pictures*.
Windows	COMMON_PROGRAMS	The file system directory that contains the directories for the common program groups that appear on the Start menu for all users. A typical path is *C:\Documents and Settings\All Users\Start Menu\Programs*. Valid only for Windows NT systems.
Windows	COMMON_STARTMENU	The file system directory that contains the programs and folders that appear on the Start menu for all users. A typical path is *C:\Documents and Settings\All Users\Start Menu*. Valid only for Windows NT systems.
Windows	COMMON_STARTUP	The file system directory that contains the programs that appear in the Startup folder for all users. A typical path is *C:\Documents and Settings\All Users\Start Menu\Programs\Startup*. Valid only for Windows NT systems.
Windows	COMMON_TEMPLATES	The file system directory that contains the templates that are available to all users. A typical path is *C:\Documents and Settings\All Users\Templates*. Valid only for Windows NT systems.
Windows	COMMON_VIDEO	The file system directory that serves as a repository for video files common to all users. A typical path is *C:\Documents and Settings\All Users\Documents\My Videos*.
Windows	COOKIES	The file system directory that serves as a common repository for Internet cookies. A typical path is *C:\Documents and Settings\username\Cookies*.
Windows	DESKTOP	The virtual folder representing the Windows desktop, the root of the namespace.
Windows	DESKTOPDIRECTORY	The file system directory used to physically store file objects on the desktop (not to be confused with the desktop folder itself). A typical path is *C:\Documents and Settings\username\Desktop*.

Group	Name	Description
Windows	FAVORITES	The file system directory that serves as a common repository for the user's favorite items. A typical path is *C:\Documents and Settings\username\Favorites*.
Windows	FONTS	A virtual folder containing fonts. A typical path is *C:\Windows\Fonts*.
Windows	HISTORY	The file system directory that serves as a common repository for Internet history items.
Windows	INTERNET_CACHE	The file system directory that serves as a common repository for temporary Internet files. A typical path is *C:\Documents and Settings\username\Local Settings\Temporary Internet Files*.
Windows	MYDOCUMENTS	Virtual folder representing the My Documents desktop item.
Windows	MYMUSIC	The file system directory that serves as a common repository for music files. A typical path is *C:\Documents and Settings\User\My Documents\My Music*.
Windows	MYPICTURES	The file system directory that serves as a common repository for image files. A typical path is *C:\Documents and Settings\username\My Documents\My Pictures*.
Windows	MYVIDEO	The file system directory that serves as a common repository for video files. A typical path is *C:\Documents and Settings\username\My Documents\My Videos*.
Windows	NETHOOD	A file system directory containing the link objects that may exist in the My Network Places virtual folder. It is not the same as *CSIDL_NETWORK*, which represents the network namespace root. A typical path is *C:\Documents and Settings\username\NetHood*.
Windows	PERSONAL	The virtual folder representing the My Documents desktop item. This is equivalent to MYDOCUMENTS.
Windows	PRINTHOOD	The file system directory that contains the link objects that can exist in the Printers virtual folder. A typical path is *C:\Documents and Settings\username\PrintHood*.

Group	Name	Description
Windows	PROFILE	The user's profile folder. A typical path is *C:\Documents and Settings\username*. Applications should not create files or folders at this level; they should instead place data under the locations referred to by *APPDATA* or *LOCAL_APPDATA*.
Windows	PROGRAM_FILES	The Program Files folder. A typical path is *C:\Program Files*.
Windows	PROGRAM_FILES_COMMON	A folder for components that are shared across applications. A typical path is *C:\Program Files\Common*. Valid only for Windows NT, Windows 2000, and Windows XP systems.
Windows	PROGRAMS	The file system directory that contains the user's program groups (which are themselves file system directories). A typical path is *C:\Documents and Settings\username\Start Menu\Programs*.
Windows	RECENT	The file system directory that contains shortcuts to the user's most recently used documents. A typical path is *C:\Documents and Settings\username\My Recent Documents*. To create a shortcut in this folder, use *SHAddToRecentDocs*. In addition to creating the shortcut, this function updates the shell's list of recent documents and adds the shortcut to the My Recent Documents submenu of the Start menu.
Windows	SENDTO	The file system directory that contains Send To menu items. A typical path is *C:\Documents and Settings\username\SendTo*.
Windows	STARTMENU	The file system directory containing Start menu items. A typical path is *C:\Documents and Settings\username\Start Menu*.
Windows	STARTUP	The file system directory that corresponds to the user's Startup program group. The system starts these programs whenever any user logs in to Windows NT or starts Windows 95. A typical path is *C:\Documents and Settings\username\Start Menu\Programs\Startup*.
Windows	SYSTEM	The Windows System folder. A typical path is *C:\Windows\System32*.

Group	Name	Description
Windows	TEMPLATES	The file system directory that serves as a common repository for document templates. A typical path is *C:\Documents and Settings\username\Templates*.
Windows	WINDOWS	The Windows directory or SYSROOT. This corresponds to the *%windir%* or *%SYSTEMROOT%* environment variables. A typical path is *C:\Windows*.

Network

Function	Description
uHostname	Returns the local network name
uSMTP	Sends a e-mail message to an SMTP server

uHostname

Description — Returns the local network name

Syntax — string uHostname()

Examples

```
uHostname()  // returns something like "pollux" or
"castor"
```

uSMTP

Description — Sends a mail to a SMTP server

Syntax — bool uSMTP(serverURL, sender, recipients, subject, body)
bool uSMTP(sender, recipients, subject, body)
bool uSMTP(recipients, subject, body)bool uSMTP(subject, body)
bool uSMTP(body)

Parameters — *string serverURL*
URL for specifying the SMTP server, port, user name, and password to use

string sender
E-mail address of sender

string recipients
Comma-separated list of recipients

string subject
Subject of the e-mail message

string body
Content of the e-mail message

Examples

```
uSMTP("Just a mail")

uSMTP("Testmail!", "Just a mail")
```

Usage

The uSMTP function allows e-mail messages to be sent to multiple recipients using a SMTP server.

The server URL for specifying the SMTP server uses this syntax:
`protocol://user:password@server:port`

Protocol can be one of:

<empty> – SMTP with SSL encryption, if applicable

SMTP – SMTP without SSL encryption

SMTPS – SMTP with SSL encryption

User name and password – The user name and password are used to authenticate the client. If not provided, no authentication is performed.

Note If the user name contains a @ sign, replace it with # to avoid ambiguities.

Port – TCP port to be used, default is 25.

```
myServer
myServer:123
SMTPS://myServer:123
Me:secret@myServer
```

Specify recipients by adding a list of addresses separated by a comma. By default, all recipients are addressed directly. To send a carbon copy or blind carbon copy, simply add "cc:" or "bcc:" before the e-mail address:

```
user@host.domain
My Name <user@host.domain>
To: My Name <user@host.domain>
To: user@host.domain
```

```
Cc: My Name <user@host.domain>
To: user@host.domain, Bcc: Test User
<test@myserver.com>
```

If the SMTP server allows encrypted communication, it is performed automatically. If you provide user name and password, authentication methods are tried in the following sequence: PLAIN, LOGIN.

You can specify your personal defaults in the *INI* file:

```
[SMTP]
ServerURL=<your default server URL>
Sender=<your default sender>
Recipients=<your default recipients>
Subject=<your default subject>
```

For example:

```
[SMTP]
ServerURL=maxm:secret@mail.gmail.com
Sender= Maxi <Max.Mustermann@ gmail.com>
Recipients=ETLAdmin@MyCompany.com, Cc: QA
qa@MyCompany.com
Subject=ETL Message
```

Numeric

Function	Description
uAbs	Returns the magnitude of a real number, ignoring its positive or negative sign
uCeil	Returns the least integer that is greater than or equal to the argument
uDiv	Returns the division integer
uExp	Returns the exponential, base e
uFloor	Returns the largest integer that is less than or equal to the argument
uLn	Returns the natural logarithm (base e) of a number
uLog	Returns the logarithm of a number
uMod	Returns the modulo of a division
uPow, uPower	Returns the value of a base expression raised to a specified power
uRandom	Returns a random number
uRound	Returns the rounded argument to the nearest integer

Function	Description
uSgn	Returns the sign of a given value
uSqrt	Returns the square root of a given value

uAbs

Description — Returns the magnitude of a real number, ignoring its positive or negative sign

Syntax — number uAbs(value)

Parameters — *number value*
A number to calculate on

Examples

```
uAbs(1522)      // returns 1522
uAbs('-123.45') // returns 123.45
uAbs('123ABC')  // returns 0
```

uCeil

Description — Returns least integer greater than or equal to argument

Syntax — number uCeil(value)

Parameters — *number value*
A number to calculate on

Examples — Round up numbers:

```
uCeil(1523.1) // returns 1524
uCeil(1523.9) // returns 1524
```

uDiv

Description — Returns the division integer

Syntax — number uDiv(value, divisor)

Parameters

number value
A number to calculate on

number divisor
The divisor of the division

Examples

```
uDiv(10, 3) // returns 3
```

uExp

Description: Returns the exponential, base e

Syntax: number uExp(value)

Parameters

number value
A number to calculate on

Examples

```
uExp(1) // returns "2.718281828459045"
```

uFloor

Description: Returns greatest integer less than or equal to argument

Syntax: number uFloor(value)

Parameters

number value
A number to calculate on

Examples

```
uFloor(1523.1) // returns 1523
uFloor(1523.9) // returns 1523
```

uLn

Description: Returns the natural logarithm (base e) of a number

Syntax: number uLn(value)

Parameters

number value
A number to calculate on

Examples

```
uLn(2.718281828) // returns 0.999999
```

uLog

Description: Returns the logarithm of a number

Syntax: number uLog(value [, base])

Parameters:

number value
A number to calculate on

number base (optional)
The base for the logarithm. If omitted, a base of 10 is used.

Examples

```
uLog(100) // returns 2

uLog(16, 2) // returns 4
```

uMod

Description: Returns the modulo of division

Syntax: number uMod(value, divisor)

Parameters:

number value
A number to calculate on

number divisor
The divisor of the division

Examples

```
uMod(10, 3) // returns 1
```

uPow, uPower

Description: Returns the value of a base expression taken to a specified power

Syntax: number uPow(value, exponent)

Parameters:

number value
A number to calculate on

number exponent
A number to be used as the exponent

Examples

```
uPow(10, 3) // returns 1000
```

uRandom

Description: Returns a random number

Syntax: number uRandom()

Examples: Random numbers

```
uRandom() // returns a value like "0.696654639123727"
```

uRound

Description: Returns the rounded argument to nearest integer

Syntax: number uRound(value [, scale])

Parameters

number value
A number to calculate on

number scale (optional)
Number of digits

Examples

```
uRound(10.1) // returns "10"
uRound(10.49) // returns "10"
uRound(10.5) // returns "11"
uRound(10.9) // returns "11"
uRound(1.235, 2) // returns "1.24"
```

uSgn

Description: Returns the sign of a given value

Syntax number uSgn(value)

Parameters *number value*
A number to calculate on

Examples
```
uSgn(-10.4) // returns -1
uSgn(0)     // returns 0
uSgn(10.4)  // returns 1
uSgn(null)  // returns null
```

uSqrt

Description Returns the square root of a given value

Syntax number uSqrt(value)

Parameters *number value*
A number to calculate on

Examples
```
uSqrt(25)   // returns 5
uSqrt(0)    // returns 0
uSqrt(null) // returns null
```

Script

Function	Description
uEvaluate	Evaluates a function or JavaScript expression and returns the result

uEvaluate

Description Evaluates a function or JavaScript expression and returns the result

Syntax string uEvaluate(expression)

Parameters *Number expression*
JavaScript code to evaluate

Examples

Evaluate functional expressions:

```
uEvaluate("3 + 5")

uEvaluate("parseFloat(IN.Salary) + 1500")
```

Define custom functions:

```
uEvaluate("function timesTwo(a){ return a*2; }")
```

Use custom functions:

```
uEvaluate("timesTwo(4)")

uEvaluate("timesTwo(IN.Salary)")
```

Evaluate scripts:

```
uEvaluate("if ("parseFloat(IN.Salary) > 2000") {2000;}
else {("parseFloat(IN.Salary) + 500");}")
```

String

Function	Description
uAsc, uUnicode	Returns the Unicode character value of a specified character
uChr, uUniChr	Returns the Unicode string corresponding to the given number, or formats a string
uCap	Returns the capitalized representation of a string
uCon, uConcat	Concatenates all given parameters into a single string
uJoin	Concatenates a delimited string with special null and empty value handling
uLeft	Returns the leftmost N characters from a string
uLength, uLen	Returns the length of a string
uSubstr, uMid	Returns a part of a string
uLPos	Finds the first position of a substring within a string
uLower, uLow	Returns the input string in lowercase letters
uLStuff	Fills the left side of a string up to a specified length
uLTrim	Removes characters from the left side of the string
uRepeat	Returns the given string repeated N times
uReplace	Replaces parts of a string
uReverse	Reverses a string
uRight	Returns the rightmost N characters from a string
uRPos	Finds the last position of a substring within a string

Function	Description
uRStuff	Fills the right side of a string up to a specified length
uRTrim	Removes characters from the right side of the string
uTrim	Removes characters from both sides of the string
uUpper, uUpp	Returns the input string in uppercase letters

uAsc, uUnicode

Description — Returns unicode character value of a specified character.

Syntax — number uAsc(value [, index])

Parameters

string value
An input string

number index (optional)
Character position for reading ASCII value

Examples

```
uAsc("Big Ben")    // returns 66
uAsc("Big Ben", 2) // returns 105
```

uChr, uUniChr

Description — Returns the Unicode string corresponding to the given number, or formats a string

Syntax — string uChr(params, ...)

Parameters

params
A list of expressions or values

Examples

```
uChr(64) // returns "@"
uChr("\u0064\u006f\u0067") // returns "dog"
uChr(65,"pple") // returns "apple"
```

uCap

Description: Returns the capitalized representation of a string. In other words, the first letter of each word in the string is capitalized.

Syntax: string uCap(text)

Parameters:

Input text
The string to be capitalized

Examples:

```
uCap('fArmeR, ASTROnaut') // returns 'Farmer,
Astronaut'

uCap('the first weekend') // returns 'The First Weekend'
```

uCon, uConcat

Description: Concatenates all given parameters into a single string

Syntax: string uConcat(params)

Parameters:

params
A list of expressions or values of any data type

Examples:

```
uConcat("For ", 3, " years.") returns "For 3 years."
```

uJoin

Description: Concatenates a delimited string with special null and empty value handling

Syntax: string uJoin(delimiter, allowEmpty, params, ...)

Parameters:

string delimiter
Delimiter to be used between all other string parts

number allowEmpty
Flag (0/1) that indicate whether empty fields are allowed

string params
List of strings to concatenate

Examples:

```
uJoin("-", 1, "James", "", "Tiberius", "Kirk") //
returns "James--Tiberius-Kirk"
```

```
uJoin("-", 0, "James", "", "Tiberius", "Kirk") //
returns "James-Tiberius-Kirk"
```

uLeft

Description — Returns the leftmost N characters from a string

Syntax — string uLeft(input, chars)

Parameters

string input
The input string

number chars
The number of characters to be retrieved

Examples

```
uLeft("James T. Kirk", 5) // returns "James"
uLeft(null, 5)            // returns null
```

uLength, uLen

Description — Returns the length of a string

Syntax — number uLength(input)

Parameters

string input
The input string

Examples

```
uLength("James T. Kirk") // returns 13
```

uSubstr, uMid

Description — Returns a part of a string

Syntax — string uSubstr(input, position, length)

Parameters

string input
Input string

number position
The position from where to start reading

number length
The number of characters to read

Examples

```
uSubstr("James T. Kirk", 7, 2) // returns "T."
```

uLPos

Description: Find the first position of a substring within a string. A result of zero indicates that the substring has not been found.

Syntax: string uLPos(input, substring)

Parameters

string input
The input string

string substring
The substring to search

Examples

```
uLPos("James T. Kirk", "T") //returns 7
```

uLower, uLow

Description: Returns the input string in lower case letters

Syntax: string uLower(input)

Parameters

string input
The string to convert

Examples

```
uLower("James T. Kirk") // returns "james t. kirk"
```

uLStuff

Description: Fills the left side of a string up to specified length

Syntax: string uLStuff(input, length [, stuff])

Parameters

string input
The string to stuff

number length
New length of string

string stuff (optional)
String to append, default is an empty space (ASCII 32)

Examples

```
uLStuff("3.5", 5)      // returns "  3.5"
uLStuff("3.5", 5, "0") // returns "003.5"
```

uLTrim

Description

Removes characters from the left side of the string. If the second parameter is omitted, it defaults to a space character (ASCII 32).

Syntax

string uLTrim(input, trimstring)

Parameters

string input
The string to be trimmed

string trimstring
The string to trim

Examples

```
uLTrim(" 3.5")       // returns "3.5"
uLTrim("003.5", "0") // returns "3.5"
```

uRepeat

Description

Returns the given string repeated N times

Syntax

string uRepeat(input, repeats)

Parameters

string input
The string to be repeated

number repeats
The number of times to repeat the input string

Examples

```
uRepeat("Hello ", 4) // returns "Hello Hello Hello Hello
"
```

uReplace

Description: Replaces parts of a string

Syntax: string uReplace(input, search, replace)

Parameters:

string input
The string to be worked on

string search
The pattern to be searched

string replace
The string to replace any match

Examples:

```
uReplace("At four o' clock he became four", "four", "4")
// returns "At 4 o' clock he became 4"
```

uReverse

Description: Reverses a string

Syntax: string uReverse(input)

Parameters:

string input
The string to reverse

Examples:

```
uReverse("Smith")  // returns "htimS"
```

uRight

Description: Returns the rightmost N characters from a string

Syntax: string uRight(input, chars)

Parameters:

string input
The input string

number chars
The number of characters to be read

Examples:

```
uRight("James T. Kirk", 4) // returns "Kirk"

uRight(null, 5) / / returns null
```

uRPos

Description: Find the last position of a substring within a string

Syntax: string uRPos(input, substring)

Parameters:

string input
The input string

string substring
The substring to find

Examples: Find the last ocurrence of a substring:

```
uRPos("James T. Kirk", "T") //returns 7
```

uRStuff

Description: Fills the right side of a string up to specified length

Syntax: string uRStuff(input, length [, stuffstring])

Parameters:

string input
The input string

number length
The new length of the result string

string stuffstring (optional)
The string to append

Examples:

```
uRStuff("3.5", 5)      // returns "3.5  "
uRStuff("3.5", 5, "0") // returns "3.500"
```

uRTrim

Description: Removes characters from the right side of the string

Syntax: string uRTrim(input [, trimstring])

Parameters:

string input
The input string

string trimstring (optional)
The string to trim

Examples

```
uRTrim("3.5 ")        // returns "3.5"
uRTrim("3.500", "0") // returns "3.5"
```

uTrim

Description: Removes characters from both sides of the string

Syntax: string uTrim(input [, trimstring])

Parameters

string input
The input string

string trimstring (optional)
The string to trim

Examples

```
uTrim(" 3.5 ")         // returns "3.5"
uTrim("003.500", "0") // returns "3.5"
```

uUpper, uUpp

Description: Returns the input string in upper case letters

Syntax: string uUpper(input)

Parameters

string input
The input string

Examples

```
uUpper("James T. Kirk") // returns "JAMES T. KIRK"
```

Trigonometric

Function	Description
uAcos	Returns the arccosine (in radians) of a number
uAsin	Returns the arcsine (in radians) of a number

Function	Description
uAtan	Returns the arctangent (in radians) of a number
uCos	Returns the cosine (in radians) of a number
uSin	Returns the sine (in radians) of a number
uTan	Returns the tangent (in radians) of a number

uAcos

Description: Returns the arccosine (in radians) of a number

Syntax: number uAcos(value)

Parameters:

number value
The input value

uAsin

Description: Returns the arcsine (in radians) of a number

Syntax: number uAsin(value)

Parameters:

number value
The input value

uAtan

Description: Returns the arctangent (in radians) of a number

Syntax: number uAtan(value)

Parameters:

number value
The input value

uCos

Description: Returns the cosine (in radians) of a number

Syntax: number uCos(value)

Parameters:

number value
The input value

uSin

Description: Returns the sine (in radians) of a number

Syntax: number uSin(value)

Parameters:

number value
The input value

uTan

Description: Returns the tangent (in radians) of a number

Syntax: number uTan(value)

Parameters:

number value
The input value

APPENDIX B

Connection Parameters

This appendix describes the database configuration options and provides additional information for some of the supported interfaces.

Interface-specific database options

In the interface-specific database options table:

- A hyphen (-) indicates that the database option has no default value. You can enter an appropriate value.
- "Not available" indicates that the database option is not provided for the underlying interface.
- "Not used" indicates that the database options is not used by the underlying interface, though it is displayed.

DB option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Always use logon credentials	Not available	0	Not available	Not available	Not available	Not available	When building the ODBC connection string, always add the credentials to the connection string.
API trace	Not available	Not available	False	False	Not available	Not available	Enable CTLIB trace facility.
API version	Not available	Not available	150	125	Not available	Not available	CTLIB API version compatibility.
Auto vacuum	Not available	Not available	Not available	Not available	Not available	0	Reclaims the space when objects are deleted from the database.

DB option	Interface and default value						Description
	Oracle	**ODBC/DB2**	**Sybase 15**	**Sybase**	**OLEDB**	**SQLite Persistent**	
BLOB chunk size	1024 While fetching LOB data from the file, ETL fetches 1024 bytes from the file and writes them to the database each time.	Not available	Not available	Not available	Not available	Not available	Determines the size at which LOBs are truncated.
BLOB fetch mode	LOB_ INLINE	INLINE	Not available	Not available	Not available	Not available	BLOB data is written either to the secondary file or held in the memory. If set to INLINE, data is held in memory. If set to FILE, data is written temporarily to the disk.
Busy timeout	Not available	Not available	Not available	Not available	Not available	10	Creates a handler, which waits for the specified number of seconds on encountering a locked database table.
Cache size	Not available	Not available	Not available	Not available	Not available	3000	Number of pages to use in cache.
CLIENT_ CHARSET	Not available	Not available	-	-	Not available	Not available	User defined character set to use with Client Library (CTLIB).
CLIENT_ CONVERSION	Not available	Not available	0 (ASE) 1 (ASA/IQ)	0 (ASE) 1 (ASA/IQ)	Not available	Not available	Controls whether or not the client library should convert data to the appropriate form.
Connect timeout	0 (Not used)	0 (Not used)	0	0	10	0	Stops trying to connect after the number of Connect timeout seconds. If set to 0, the connect does not timeout.
CONVERTER_CHARSET	Not available	Not available	-	-	Not available	Not available	The character set to be used when CLIENT_ CONVERSION = 1.
Database name	Not available	Not available	-	-	Not available	Not available	Database name.
DBMS_VER	Not available	Not available	-	-	Not available	Not available	Database version.

DB option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Default cache size	Not available	Not available	Not available	Not available	Not available	3000	Default number of pages to use in cache.
Disconnect timeout	Not available	10 On Windows 32-bit, ETL always uses the default value. On other platforms, this option is not used.	Not available	Not available	Not available	Not available	Enforces disconnection from the database, if there is no reply from the database for *n* seconds after you try to disconnect.
Enable SQL Server fast load	Not available	Not available	Not available	Not available	1	Not available	If set to 1, the MS SQL Server fast load feature is enabled. If set to 0, the feature is disabled.
Execution timeout	0 (Not used)	0 /Not used	-1	-1	Not available	0	Component stops execution after a time interval in seconds. (0 <= means no timeout).
Extended connect options	Not available	-	Not available	Not available	-	Not available	Allows additional driver-specific parameters to be added to an ODBC connection string.
Full column names	Not available	Not available	Not available	Not available	Not available	1	When set to 1, column names are fully qualified, following this pattern: <table-name/alias> <column-name>.
Internal database	Not available	Not available	Not available	Not available	Not available	-	Database reference.
Isolation level	DEFAULT (Not used)	DEFAULT	DEFAULT (Not used)	DEFAULT (Not used)	Not available	DEFAULT (Not used)	Defines the degree to which one transaction must be isolated from resource or data modifications made by other transactions.
Lock resultset data	0 (Not used)	0	0 (Not used)	0 (Not used)	Not available	0	Query tables will be locked ensuring that no data is written to the selected record set while the process is working on it. The selected record set is released when the last record from that set is fetched.

DB option	Interface and default value						Description
	Oracle	**ODBC/DB2**	**Sybase 15**	**Sybase**	**OLEDB**	**SQLite Persistent**	
Log SQL statements to a file	0	0	0	0	0	0	If set to 1, all SQL statements are logged to the log or *SQL.log* file.
Numeric support	Not available	0 This value is 0 rather than the user input, when DBMS is IQ or ODBC driver is ASA9.	Not available	Not available	Not available	Not available	Whether or not to enable ODBC numeric support.
Object name end quote	"	- ODBC uses the value queried from DBMS rather than the user input.	]	]	-	Not available	When creating SQL statements, terminating character are used as quotes.
Object name start quote	"	"" ODBC uses the value queried from DBMS rather than the user input.	[	[	-	Not available	Beginning character to be used as quote when building SQL statements.
PAD_BLANKS	Not available	Not available	0	0	Not available	Not available	Maintains a constant column width using space characters.
Page size	Not available	Not available	Not available	Not available	Not available	4096	Number of bytes per page. Must be a power of 2, greater than or equal to 512, and not higher than 32768.
Quote character	"	Not available	Not available	Not available	Not available	Not available	Same as QUOTE_START and QUOTE_END.
Quote object names	0 (Not used)	0	0	0	0	0	If set to 1, the character specified in QUOTE_START and QUOTE_END is used to surround identifiers in generated SQL statements.
Reject log column delimiter	tab	tab	tab	tab	-	tab	Used as a column delimiter in the reject log.

DB option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Short column names	Not available	Not available	Not available	Not available	Not available	0	If flag is set to false, column names are fully qualified, otherwise they are referred to only by the column name.
Show all tables	Not available	Not available	0 (Not used)	0 (Not used)	Not available	Not available	Display system tables as well as user tables.
Show error location	1	1	1	1	1	1	Display the error location if set to 1. Database errors include the position of the record within the result set.
SHOW_ ERROR_ LOCATION_ ABSOLUTE_ ROWS	1	1	1	1	1	1	Show the error location from the absolute begining of the result set, rather than the current result set.
Synchronous	Not available	Not available	Not available	Not available	Not available	0	If you select Full =2, SQLite ensures data is written to disk before continuing. If you select Normal=1, SQLite pauses to write at critical moments but not as frequently as when set to 2. If you select Off = 0, data is handed off to operating system and SQLite continues.
Temp store	Not available	Not available	Not available	Not available	Not available	2	If set to 1, the location of the temporary database is a file. If set to 2, the location of the temporary database is memory.
Treat numeric value as character	Not available	1 If database is IQ, or driver name contains “SYSYBNT” or “LIBDB2.A,” ODBC uses 1 instead of the user input.	Not available	Not available	Not available	Not available	Force conversion of numeric data to string.

DB option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Use system views	True	Not available	Not available	Not available	Not available	Not available	Use DBA system tables to show metadata instead of per-user metadata.
Validate result column binding	Not available	Not available	Not available	Not available	1	Not available	If set to 1, the result column mapping binding is validated when reading data from the database.
Write empty dates as NULL	0	0	0 (Not used)	0 (Not used)	Not available	Not available	If set to 1, the value of empty dates are enforced to be NULL.
Write rejected records to file	-	-	-	-	-	-	Specifies the file path for reject logs. This option is used to log records that are rejected by the database on loading.

Database and interface support

The following table lists the interfaces and databases available for each database component type.

Table B-1: Database and interface support matrix

Interface	DB Data Provider and DB Lookup	DB Data Sink	DB Bulk Load Sybase IQ	DB Staging	IQ Loader File via Load Table	IQ Loader DB via Insert Location	SQL Executor
Sybase	ASE IQ	IQ	IQ	ASE ASA IQ	IQ	IQ	ASE IQ
ODBC	Any data source that can be accessed via ODBC.	IQ	IQ	IQ ASA ASE	IQ	Not supported	ASE IQ SQL Server Oracle
OLE DB	SQL Server	Not supported	Not supported	Not supported	Not supported	Not supported	SQL Server

Interface	DB Data Provider and DB Lookup	DB Data Sink	DB Bulk Load Sybase IQ	DB Staging	IQ Loader File via Load Table	IQ Loader DB via Insert Location	SQL Executor
Oracle	Any Oracle database system that can be accessed by Oracle Call Interface (OCI).	Not supported	Not supported	Not supported	Not supported	Not supported	Oracle
DB2	Any DB2 database system that can be accessed by the IBM DB/2 client interface.	Not supported	Not supported	Not supported	Not supported	Not supported	DB2

The Sybase ETL environment has been tested, evaluated, and verified thoroughly to comply with many interface drivers of the supported database systems.

If you encounter unexpected results that might be related to driver incompatibility, try installing one of the supported versions for your interfacing driver. See “Interface support” in the *Sybase ETL 4.9 Release Bulletin* for a list of all interface driver versions supported by Sybase ETL 4.9.

Working with the SQLite Persistent interface

Sybase ETL technology includes a built-in, general purpose, relational database you can use for temporary data storage and staging. It is based on SQLite, a very fast, widely used, mostly SQL92-compliant database. SQLite is a small C library that implements a self-contained, embeddable, zero configuration SQL database engine.

Connecting to a SQLite database

A SQLite database is represented as a single file with the *.db* extension. The database file can contain any number of tables.

❖ **Creating or connecting to a SQLite database file**

1 In the Properties window, select SQLite Persistent from the Interface menu.

2 Provide the host name for the SQLite database file.

- To create a new SQLite database file, provide a name in the Host Name field; do not include the *.db* extension. A new SQLite database file, with the extension *.db*, is automatically created in the default location, which is the *database* directory under the installation folder.

  To create the SQLite database file in a directory other than the default, specify the complete path, including the *.db* extension, in the Host Name field.
- If you are connecting to an existing SQLite database file in the default directory, select the file name from the Host Name menu. Do not enter the *.db* extension. To connect to a SQLite database file in a directory other than the default, specify the complete path, including the *.db* extension, in the Host Name field.

For example, to create a new SQLite database file called *mySQLite.db*, or to connect to an existing *mySQLite.db* database file, use these parameters:

- Interface: SQLite Persistent
- Host Name: mySQLite

Creating a SQLite table

Create a SQLite table in one of these ways:

- Right-click a Staging component and select Create Staging Table from Input or Create Staging Table from Port.
- Right-click one of the Data Sink components and select Add Destination Table from Input or Add Destination Table from Port.

Extracting data from a SQLite database

Provide the proper connection parameters for the SQLite database file on a DB component.

You can use SQLite-supported SQL commands in the preprocessing or postprocessing SQL properties of components connected to databases.

Use the Content Explorer from the Tools menu to manipulate or browse objects of the SQLite database connected to components in your project. You can also use client applications available from *sqlite.org* to connect to SQLite database files.

Note To use external client applications to connect to your SQLite database files, you must be familiar with the locking strategy of SQLite.

Working with the Oracle interface

Table B-2 lists class-level translations for Sybase ETL datatypes to Oracle datatypes.

Table B-2: Data type mapping from Oracle interface to Sybase ETL

ETL datatype	Oracle datatype	Size/precision	Minimum scale	Maximum scale
binary	BLOB	2147483647		
binary	BFILE	2147483647		
binary	RAW	2000	0	0
string	CLOB	2147483647	0	0
string	CHAR	2000	0	0
float	DECIMAL	38	0	0
integer	NUMBER	38	0	0
float	DOUBLE PRECISION	15	0	38
datetime	DATE	19	0	0
datetime	TIMESTAMP	28	0	9
string	VARCHAR2	4000	0	0
unicode	NCHAR	1000	0	0
unicode	NVARCHAR2	2000	0	0
unicode	NCLOB	2147483647	0	0

APPENDIX C

Using ETL for Slowly Changing Dimensions

This chapter provides an overview of slowly changing dimensions (SCDs). It lists some common SCD scenarios and describes how these scenarios are implemented using ETL projects and jobs.

Overview

Slowly changing dimension is a common data warehousing scenario. SCD utilizes three different method types for handling changes to columns in a data warehouse dimension table.

Type 1

In Type 1, new data overwrites existing data. The existing data is lost and there is no tracking of historical changes. Type 1, is the easiest method to support, but is useful only if you need not track historical changes.

Consider a table that keeps product information:

Key	Name	Price
1	Notebook	1200

The price of the notebook increases to 1500. The updated table simply overwrites the current record:

Key	Name	Price
1	Notebook	1500

Type 2

Type 2 retains the full history of values. If new data differs from the old data, an additional dimension record is created with new data values and becomes the current record. Each record contains the effective date and an expiration date to identify the time period for which the record was active. Use type 2 to keep a full history of dimension data in the table.

See “Case study scenario” on page 310.

Type 3

Type 3 tracks changes using separate columns. There is one version of the dimension record that stores the previous value and current value of selected attributes. Use type 3 when you need to track historical changes that occur only for a finite amount of time.

Consider a table that keeps product information:

Key	Name	Price
1	Notebook	1200

The price of the notebook increases to 1500 on 15th July 2008. To accommodate type 3, new columns are added, Current Price and Effective Date:

Key	Name	Original price	Current price	Effective date
1	Notebook	1200	1500	2008-07-15

Note Type 3 is rarely used because altering the structure of the dimension table should be undertaken for only a very important change.

Case study scenario

This section provides a case study scenario for type 2 SCD and describes how to create transformation projects in Sybase ETL to implement this scenario.

Case description

You have two tables:

- PRODUCT in the operational or source database.
- PRODUCT_PRICE in the data warehouse or target database. This table tracks modification of products in the source table (PRODUCT) over time, such as:
  - Change in price of existing products

- Newly added products
- Deleted products

The database table schema of the PRODUCT table looks like:

Column	Description
Key	Unique ID of product
Name	Name of the product
Price	Price of the product

The database table schema of the PRODUCT_PRICE table looks like:

Column	Description
Key	Source key identifier in the source table (PRODUCT)
Name	Name of the product
Price	Price of the product
Valid From	Date of insertion of new records
Valid To	End of validity of records. A record becomes invalid when a new record with the same source key is inserted in the PRODUCT_PRICE table.

Rules

The rules to transfer dimensions from the PRODUCT table to the PRODUCT_PRICE table are:

1 If the record does not exist in the PRODUCT_PRICE table, create it with the same column values as the PRODUCT table. Set the Valid From date to the record insertion date, and the Valid To date to 9999-12-31.

2 If the record exists in the PRODUCT_PRICE table, but nothing has changed, do not insert a new record or update an existing one.

3 If the record exists in the PRODUCT_PRICE table, and the price of the record has changed:

   - Set the Valid To date for the record with the old price to yesterday.
   - Create a new record. Set the Valid From date to the record insertion date, and the Valid To date to 9999-12-31.

4 If the record exists in the PRODUCT_PRICE table but has been deleted from the PRODUCT table, set the Valid To date of the product in the PRODUCT_PRICE table to yesterday.

Note A running history of dimension changes, based on these rules, is maintained in the PRODUCT_PRICE table.

How it works

After initial load on 01 January 2008, the PRODUCT table displays:

Key	Name	Price
1	Notebook	1200
2	Monitor	1000
3	Mouse	500

After the ETL process is run for the first time, the PRODUCT_PRICE table displays:

Key	Name	Price	Valid from	Valid to
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	9999-12-31
3	Mouse	500	2008-01-01	9999-12-31

On 15 January 2008, the PRODUCT table is updated when the price of the monitor is modified.

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
3	Mouse	500

After the ETL process is run again, the PRODUCT_PRICE table displays:

Key	Name	Price	Valid from	Valid to
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	9999-12-31
2	Monitor	1400	2008-01-15	9999-12-31

On 22 January 2008, the PRODUCT table is updated again when a new product, a hard disk, is added.

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
3	Mouse	500
4	Hard Disk	1000

After the ETL process is run again, the PRODUCT_PRICE table displays:

Key	Name	Price	Valid from	Valid to
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	9999-12-31
2	Monitor	1400	2008-01-15	9999-12-31
4	Hard Disk	1000	2008-01-22	9999-12-31

The PRODUCT table is updated again on 28 July 2008, to remove the mouse as an available product.

Key	Name	Price
1	Notebook	1200
2	Monitor	1400
4	Hard Disk	1000

After the ETL process is run again, the PRODUCT_PRICE table displays:

Key	Name	Price	Valid from	Valid to
1	Notebook	1200	2008-01-01	9999-12-31
2	Monitor	1000	2008-01-01	2008-01-14
3	Mouse	500	2008-01-01	2008-07-27
2	Monitor	1400	2008-01-15	9999-12-31
4	Hard Disk	1000	2008-01-22	9999-12-31

Setting up ETL projects for SCD

This section describes ETL concepts for accomplishing type 2 SCD using projects and jobs. The demo repository that is packaged with the product includes various ETL transformation objects related to the type 2 use case, including:

Projects

- Demo Product Price SCD – Initial Load

  This project initializes or reinitializes the demo environment for the SCD – Update projects and job.

  Note This project is not a part of the use case implementation. In a production environment, the first execution of the Update New and Modified project, on an empty target table, performs the initial load where all source records are processed as new records. Since the demo environment uses two different tables to simulate changes on the source data, the original data always needs to be restored in the target table by executing this project before running the other 2 update projects or the job.

- Demo Product Price SCD – Update New and Modified

  This project updates the dimension table of the target database on a daily basis to reflect modification or addition of products in the source database. See Rules 1 – 3 in "Case study scenario" on page 310.

  To accomplish a full update, also execute the SCD – Update Deleted project.

- Demo Product Price SCD – Update Deleted

  This project updates the dimension table of the target database on a daily basis to reflect deletion of products in the source database. See Rule 4 in "Case study scenario" on page 310.

  To accomplish a full update, also execute the SCD – Update New and Modified project.

Job

- Demo Product Price SCD – Daily Update

  This job executes both the SCD – Update New and Modified and SCD – Update Deleted projects, and provides a single transformation object for performing a full update of the target dimension table. Before executing this job, execute the SCD – Initial Load project.

Understanding target dimension table

Identifying target records

A target dimension table contains multiple records for the same source key. To differentiate a current version of a record from a historical version, the target dimension table uses a compound key, which includes the source key and either the effective date or the expiration date attribute. The ETL demo projects use the Valid From date attribute as part of the key.

Current target records

Each record in the source table is represented by exactly one current record in the target table. Only current records are relevant for SCD when checking the target dimension table. In the ETL demo projects the current records have a Valid To date of 9999-12-31.

Detecting source changes

This section describes how to capture changes in the source table, including new source records, modified source records, and records that have been deleted from the source. Methods can be combined to detect different types of data changes in one step.

In the case study scenario, the source database does not contain any change log information, so source and the target content must be compared to detect any changes. Since, in most cases, the source and target objects do not reside in the same database, a heterogeneous join needs to be performed.

Detecting new source records

Records that are added to the source after the last update of the target dimension table do not have a corresponding current record in the target dimension table.

1 All source records are read using an appropriate Data Provider component. For a list of Data Provider components, see “Source components” on page 95.

Note All attributes that are transferred to the target dimension table are read, although only the key attribute is required for detection.

2 The existence of a corresponding current record in the target dimension is checked for each source record based on the source key attribute.

a Choose an appropriate Lookup component. See “Lookup components” on page 149.

To perform calculations on the data to be transferred, consider using the lookup functionality of the Data Calculator component. See “Data Calculator JavaScript” on page 136.

b Select the lookup data from target. As this is a simple existence check, only the original source keys are needed from all current target records. However, lookups in ETL always return a value for a key, so you must also select an appropriate return value.

c Add an attribute to the port structure to populate the lookup result. The lookup result determines whether a source record is newly added or existed previously. This attribute indicates the data state and allows data to be filtered in the next step of the transformation process. See “Modifying port structures” on page 90. The new attribute is selected as the value attribute of a Lookup component or the output attribute in the Data Calculator.

d Set an appropriate lookup default value. The default value is returned by the lookup for nonexisting keys. To ensure that the lookup value correctly indicates the existence of records, set it to a constant that is different from all lookup values for any existing keys.

Example:

- Source data – select Key, Name, Price from PRODUCT
- Lookup data – select Key, '1' from PRODUCT_PRICE where Valid_To = '9999-12-31'
- Value attribute – Exists (integer)
- Default value – 0

- Performing this lookup results in records with attributes Key, Name, Price, Exists. The value of the Exists attribute will be 1(lookup value) for all records existing in target, and 0 (default value) for all nonexisting records.

Detecting modified source records

Records that are modified in the source after the last update of the target dimension table have a corresponding current record in the target dimension table, but the relevant values are changed.

1 All source records are read using an appropriate Data Provider component. See “Source components” on page 95.

Note All attributes that are transferred to the target dimension table are read, although only the key attribute is required for detection.

2 The existence of a corresponding current record in the target dimension table is checked for each source record based on the source key attribute, and the values are compared.

a Choose an appropriate Lookup component.

Use the Data Calculator component to look up multiple values for a single key attribute and perform comparisons. See “Data Calculator JavaScript” on page 136.

b Select the lookup data from the target. The key attributes and all values to be compared for all current target records are read using an appropriate Data Provider component. See “Source components” on page 95.

c Add an attribute to the port structure to populate the additional target key attribute. The lookup result determines whether a source record has been modified or is either new or unchanged. This attribute indicates the data state and allows data to be filtered in the next step of the transformation process. The update operation on the target must uniquely identify the current record, thus the date part of the target key needs to be populated as well. See “Modifying port structures” on page 90.

d Set an appropriate lookup default value. The default value is returned by the lookup for nonexisting keys. To ensure that the lookup value correctly indicates the existence of records, set it to a constant that is different from all lookup values for the existing keys.

e Look up all necessary target values. The first lookup uses the new target key attribute as the output attribute in the Data Calculator, thus indicating existence. The values to be compared are read into temporary variables.

f Compare source and target attribute values. The target key attribute is recalculated, based on a value comparison for the existing records.

Example:

- Source data – select Key, Name, Price from PRODUCT
- Lookup data – select Key, Valid_From, Price from PRODUCT_PRICE where Valid_To='9999-12-31'

First check existence by reading the effective date from target:

- Output attribute – Valid_From
- Default value – 0

Read current target price into temporary variable:

- Output Attribute – Tmp_Price
- Default Value – 0

If a current target record exists (Valid_From is not 0), compare Price and Tmp_Price. Recalculate Valid_From to 0 if the Price has not changed.

Performing these lookups and calculations results in records with Key, Name, Price, and Valid_From attributes. Valid_From either contains the effective date of the target record to be updated or contains 0, indicating new and unchanged records.

Detecting deleted source records

Records that have been deleted from the source after the last update of the target dimension table still have a corresponding current record in the target dimension table.

1 Key attributes of all current records in the target dimension table are read using an appropriate Data Provider component. See “Source components” on page 95.

2 The existence of a corresponding record in the source is checked for each current target record based on the source key attribute.

a Choose an appropriate Lookup component. See “Lookup components” on page 149.

If your source data does not reside in a database, use the lookup functionality of the Data Calculator component. See “Data Calculator JavaScript” on page 136.

b Select the lookup data from source. As this is a simple existence check, only the source keys are needed from all source records. However, lookups in ETL always return a value for a key, so you must select an appropriate return value as well.

c Add an attribute to the port structure to populate the lookup result. The lookup result determines whether a source record has been deleted. This attribute indicates the data state and allows the data to be filtered in the next step of the transformation process. The new attribute is selected as the value attribute of a Lookup component or the output attribute of a Data Calculator rule. See “Modifying port structures” on page 90.

d Set an appropriate lookup default value. The default value is returned by the lookup for nonexisting keys. To ensure that the lookup value correctly indicates the existence of records, set it to a constant that is different from all lookup values for the existing keys.

Example:

- Target data – select Key, Valid_From from PRODUCT_PRICE where Valid_To='9999-12-31'
- Lookup data – select Key, '0' from PRODUCT
- Value attribute – Removed (integer)
- Default value – 1

Performing this lookup results in records with attributes Key, Valid_From, Removed. The value of the Removed attribute will be 0 (lookup value) for all existing source records and 1 (default value) for all nonexisting records.

Alternatives

- If the source is a database that provides ascending indicator for insertions, updates, or deletions (like autoincrements, modification dates, and so forth), the DB Data Provider Index Load component can be used to read records changed since the last load only. See “DB Data Provider Index Load” on page 99.
- Use a Staging component to load relevant data from both the source and target to the same database. New, modified, and deleted records are then detected by extracting data from the stage using a full outer join. See “DB Staging” on page 157.

Filtering the records

Use the Data Splitter component to remove records from the data stream, apart from splitting data streams to more than one output. See “Data Splitter JavaScript” on page 143.

To remove records from the data stream, define conditions for every OUT-port, such that the records to be removed do not match any of them. To output a single data stream, configure a Data Splitter with a single OUT-port by deleting one of the default OUT-ports.

Populating the target dimension table

Assigning values to target attributes

Using the insert and update options of the DB Data Sink components, values are assigned to those target attributes that are not included in the inbound data stream. The values are constant for all records processed during a single execution but allow for SBN expressions to dynamically initialize them. See “DB Data Sink Insert” on page 180 and “DB Data Sink Update” on page 186.

In the ETL demo projects and jobs, the insert options use the values:

- Dynamic – '[uDate('now','localtime')]' (today) for the Valid From date attribute.
- Static – '9999-12-31' for the Valid To date attribute.

The update options use the dynamic value '[uDate('now','localtime','-1 day')]' (yesterday) for the Valid To date attribute.

Performing partial updates

The update options of a DB Data Sink component allow a subset of attributes to be updated, instead of updating the complete record. To exclude attributes from update, unselect them in the update options window. See “DB Data Sink Update” on page 186.

In the ETL demo projects, old records are outdated by updating the Valid To date attribute.

APPENDIX D

Best Practices

This appendix describes best practices for working with Sybase ETL.

Best practices for working with ETL Server

Avoid starting multiple ETL Server sessions

If you start ETL Server from the command line while ETL Development is running, ETL Development becomes unstable and displays error messages if you perform any action in ETL Development. This happens due to conflicts between the ETL Server session you started from the command line and the ETL Server session started by ETL Development.

To avoid starting multiple server sessions, in the Preferences window, unselect Engine | “Start local engine during application startup.” This prevents the ETL Server session from starting automatically next time you start ETL Development.

Enter the default port number for command line execution

Command line execution fails if you attempt to use the default port number of 5124 but you do not include it in the command line. To avoid this issue, you must enter the default port number of 5124 in the command line. For example:

```
GridNode -con --port 5124 --server localhost
-f "..\\testdata\\tpms\\tpms_TestB.xml"
```

Use column aliases when entering queries

Output column names for query result sets are generated by the source database. When you apply a function to an attribute in a query, the output column name is different for different databases and may bear no relation to the source column name. You can define custom column names to be used as port attribute names and displayed in the Data Viewer, by adding column aliases when you enter queries.

For example, when querying an ASE database using an aggregate function, the Content Browser displays the results of the query without a column header for the aggregated data column. To add a column header, add an alias value on the same row as the attribute, which utilizes the aggregate function. If the query is:

SELECT COUNT (qsID) FROM TAB_IDS, add an alias value to the returned

column as follows:

SELECT COUNT (qsID) AS qsID_COUNT FROM TAB_IDS

Do not perform DDL operations in transactional projects

Database target or interface combinations behave differently for DDL transactions, so avoid performing DDL operations in pre and post sql processing for transactional projects.

If needed, perform DDL operations in a non-transactional project. For example, create a job that has 3 projects:

- Non-transactional setup project (including DDL)
- Transactional project
- Non-transactional cleanup project (including DDL)

Best practices for working with ETL components

Migrating wide tables

Migrating tables with hundreds or thousands of columns consumes a lot of memory. To prevent errors while migrating wide tables with varying numbers of columns and rows from a source Sybase IQ database to a target Sybase IQ database, follow these recommendations:

- Allocate a 1GB database space for the SQL Anywhere repository with all the other values remaining as system defaults.
- Use the following techniques, components, and interfaces to migrate tables with large number of columns:

Number of columns	Migration techniques	Interface to use
Up to 3000 columns	Insert Location component	Sybase
Up to 3000 columns	Load Table component	Sybase or ODBC
Up to 3500 columns	Migration wizard **Note** The Migration wizard may fail to generate the migration projects, either due to memory limitations, or if you do not use a SQL Anywhere repository.	Sybase or ODBC
Up to 10000 columns	Load Table component	ODBC

Note When you are migrating a table with more than 3000 columns, you may encounter performance issues moving large number of rows.

Importing XML file with more than 32 sibling elements

To import more than 32 sibling elements using the XML via SQL Data Provider component, set the Create Flat View to 0 in the XML Options property of the component. You must manually set up sub queries using the Content Explorer.

To load last row of source text file to Sybase IQ

When you are using the IQ Loader File via Load Table component, Sybase IQ does not accept the last row of a source text file from ETL if the last row does not end with a trailing row delimiter. To avoid this issue, add a line delimiter at the end of the last row in the source text file. For example, in Windows, with the row delimiter specified as CRLF, place the cursor at the end of the last row and press Enter to add a row delimiter to the last row.

Configure Adaptive Server Enterprise for bulk copying

If you are using Adaptive Server Enterprise for DB Staging, you must first configure the Adaptive Server database for bulk copying. If the Adaptive Server is not configured, you may encounter errors, although the project executes successfully.

Add less than 35 Data Calculator JavaScript and DB Staging components

Make sure you do not add more than 35 Data Calculator JavaScript and DB Staging components to a single project.

Increase the text size for the Adaptive Server ODBC driver

The Adaptive Server ODBC driver truncates text or image data values that are larger than the value set in the ODBC configuration in Microsoft Windows for the driver. You must increase the text size value in the ODBC Control Panel, or set the value in the database options parameter for your database connection to avoid this issue.

❖ **Increasing the text size value using the Control Panel**

1. Select Control Panel | Administrative Tools | Data Sources (ODBC), then select the data source for Adaptive Server Enterprise in the User DSN or System DSN tab.
2. Select Configure to display the ODBC Adaptive Server Enterprise Setup window, then select Advanced.

3 Change the value for Text Size to a value larger than 32KB, which is the default. The Adaptive Server ODBC drive truncates any data value that is larger than the value you set here.

❖ **Increasing the text size value using Database Options**

1 In the Properties window for the database connection, double-click the edit icon of the Database Options field to display the Enter Properties window.

2 In the "Extended Connect Options" field, enter "TEXTSIZE=*N*" where *N* is the text size value you want to set.

Delimiters in the source text file should not change when project is executed on different platforms

When you create a project using the Text Data Provider component on Windows and then execute it on UNIX or Linux, make sure the source text file is not converted to the UNIX or Linux format. The delimiters used in the source text file should always be same as what was designed in the Text Data Provider component. If the delimiters are inconsistent, you will encounter execution errors.

Setting named pipe permission on Windows

For the DB Bulk Load Sybase IQ component, if you want to provide a pipe name in the Load Stage property field, you must first make the following settings to avoid any errors:

1 Go to Control Panel | Administrative Tools | Local Security Policy | Local Policies | Security Options.

2 Double-click "Network access: Named Pipes that can be accessed anonymously" from the Policy list.

3 Add your named pipe to the existing list. For example, if the pipe name is "pipe://mypipe", add "mypipe" to the list. Click Apply.

4 Click OK.

Migrating tables to IQ containing LOB columns

Before migrating tables to IQ that contain Character Large Object (CLOB), Binary Large Object (BLOB), image or text columns with the IQ Loader DB via Insert Location component, review your IQ configuration. The IQ parameter LOAD_MEMORY_DB controls how much system memory IQ uses to manage these types of columns. By default, IQ sets this parameter to 0, which means that the memory usage for processing these requests is unlimited. You may see this error:

```
ASA Error -1006042: All available virtual memory
has been used; allocation cancelled:
Extra info: 948472704].
```

If you set this parameter to a specific value, for example, 300 (MB), IQ uses only the specified amount of memory to process these requests, and prevents the error from occurring. For more information on IQ parameters, see Chapter 2, “Database options,” in the *Sybase IQ 12.7 Reference Manual*.

Best practices for working with internationalization

Parsing source files with byte-order mark correctly

If you are using the Fixed by Bytes property to parse your file, make sure the source file does not include the byte-order mark. Use a text editor to remove the byte-order mark from the source file before parsing it.

Set ETL to support UTF-8 encoding

Arguments given to the uSetEnv function cannot contain multibyte or non-Western characters. You must set ETL to support UTF-8. For example:

```
set LANG=zh.UTF-8
```

Select correct character set encoding to display Unicode characters properly

Characters display incorrectly unless you select the correct “endianness” type for the character set encoding for Unicode files that have a byte order marker (BOM), when you load character data using the “Text Data Provider” or “Text Data Sink” components.

To display the Unicode characters correctly, in the Character Encoding field of the component configuration window, select the character set encoding with the correct endianness type for character data. For example, select:

- UTF-16LE – to process text files encoded in UTF-16LE that have a BOM at the beginning of the file where LE means “little-endian” since the BOM is at the beginning of the file.
- UTF-16BE – to process text files encoded in UTF-16BE with a BOM at the end of the file where BE means “big-endian” since the BOM is at the end of the file.

Index

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

X