SYBASE®

Users Guide

Sybase IQ ETL

4.5.1

DOCUMENT ID: DC00608-01-0451-01

LAST REVISED: November 2008

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

Contents

About This Book

Audience

This guide is for users of Sybase® IQ ETL Development.

How to use this book

This book contains these chapters:

- Chapter 1, “Sybase IQ ETL” is an overview of the Sybase IQ ETL architecture and the feature set of Sybase IQ ETL Development and Sybase ETL Server.
- Chapter 2, “Getting Started” describes how to get started using Sybase IQ ETL. It familiarizes you with the Sybase IQ ETL Development interface and describes the functions you can perform using the interface.
- Chapter 3, “Projects and Jobs” guides you through creating, simulating, and executing 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 IQ ETL components that are used to create projects and jobs.
- Chapter 6, “Sybase ETL Server” provides infomation on how to use Sybase IQ ETL Server.
- Appendix A, “Function Reference” describes the built-in functions available in Sybase IQ ETL.
- Appendix B, “Connection Parameters” describes the database configuration options. It also 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 IQ ETL.

Related documents

See the following documents for more information:

- Sybase IQ ETL *New Features Guide* – describes the new features in Sybase IQ ETL 4.5.1.
- Sybase IQ ETL *Release Bulletin* – contains last-minute information that was too late to be included in the books.

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 IQ 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

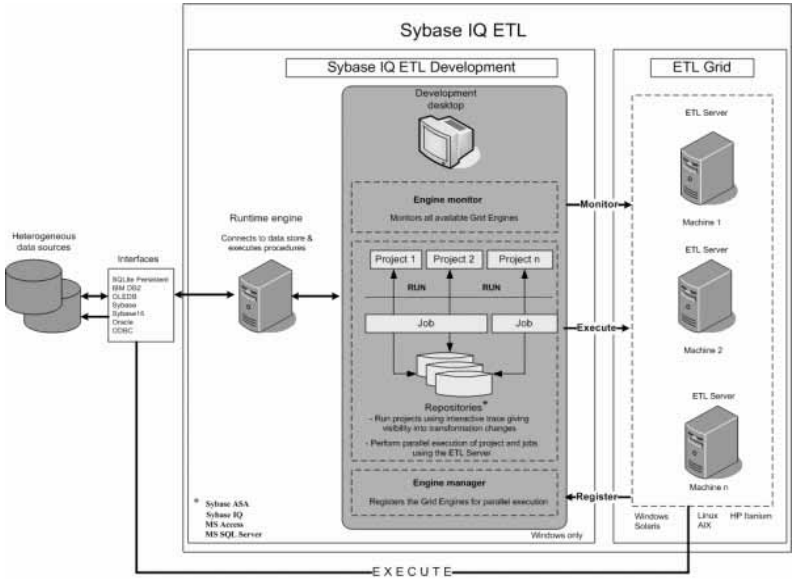
Sybase IQ ETL

Sybase IQ 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 IQ 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 IQ ETL architecture



Sybase IQ ETL includes Sybase IQ ETL Development and Sybase ETL Server.

Sybase IQ ETL Development

Sybase IQ ETL Development 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. It includes a runtime engine that controls the actual processing, such as connecting to databases and executing procedures. It is available only on Windows.

Sybase ETL Server

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 IQ ETL Development.

You can add multiple ETL servers on different operating systems within your network. Each server exposes certain services to all other peer servers. Sybase IQ ETL uses the various servers on a grid for parallel execution of projects and jobs, which improves scalability of transformation speed. See Chapter 6, “Sybase ETL Server.”

Note For parallel execution of projects and jobs developed using Sybase IQ ETL Development, you must install the Sybase ETL Server, which is available as a separate executable.

To make ETL servers available for parallel execution, register them in the Engine Manager. See “Using multiple engines to reduce job execution time” on page 77.

To monitor multiple servers on a grid, use the Engine Monitor. See “Engine Monitor” on page 79.

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

Interfaces

The interfaces are methods or drivers that you use to connect to the destination or the source database. Out of the supported interfaces, ODBC driver used for connecting to Sybase IQ is automatically installed by Sybase ETL. To install the other supported interfaces, see the respective vendor documentation.

Sybase IQ ETL concepts

This section introduces Sybase IQ ETL concepts.

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. When simulated or executed, the 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. You can add components to your project by dragging them from a section of the Component Store onto your Design window.

You can run multiple projects sequentially or parallely in a job. 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 or 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 start the simulation from the very beginning of the project after any of the components have been changed.

You can simulate your projects interactively or non interactively. Either way, data is written into the data sinks at the end of simulation. See “Simulating projects interactively” on page 30 and “Simulating projects noninteractively” on page 32 for details.

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

Execution mode enables you to:

- Run projects and jobs that have been saved to the repository. Unsaved changes are not executed.
- Directly 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 IQ ETL Development, or as a scheduled task. See “Managing jobs and scheduled tasks” on page 59.

Customizing a project

You can create data transformation projects without manually entering a single line of programming code or SQL statements. For example, you can:

- Generate select statements inside queries, lookup definitions, preprocessing and postprocessing SQL using the Query Designer.
- Map attributes between data sources and data sinks using the data mapping features of the links between the components.
- Create temporary or persistent staging tables using the built-in Create Table From Port command of the component you are using.
- Create tables in the destination database using the built-in create table from port command of the component you are using.
- 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 IQ ETL Development.

Additionally, when dealing with complex data transformation requirements, you can use:

- Manually optimized SQL select statements to adjust the data extraction process.
- SQL to apply data manipulation commands inside preprocessing and postprocessing commands.
- JavaScript to write procedures, perform complex calculations, or manipulate objects in the operating system environment.
- Indirection in expressions (Square Bracket Notation) to assign values dynamically to 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 visualize the result of any applied transformation immediately.

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 IQ 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 32.

Repositories

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

During a session, multiple repositories are accessible in parallel. You can copy and transfer projects between repositories, which allows you to separate your production repository from your development repository.

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 client users; each user has a user name and password that controls the 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 might be lost.

Datatypes and data formats

Data source datatypes are preserved during transformation.

Internally, Sybase IQ 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 do not have to 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 setting in the Preference window and manually convert the data. See “Customizing preferences” on page 21.

SQL

Most of the data delivered by Data Providers is defined by using SQL statements stored in the Query property. Sybase IQ 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.

Tools

Structural and catalog information from all connected data sources is accessible through Sybase IQ 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 53 for information on the various Sybase IQ ETL tools.

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 IQ ETL allows you to extract, transform, and load:

- Unicode characters
- These 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
  - Square Bracket Notation (SBN) expressions

Expressions

Square Bracket Notation (SBN) is a widely applicable indirection mechanism within the Sybase IQ ETL environment. You can apply Square Bracket Notation within expressions, SQL statements, and file name specifications. Use Square Bracket Notation to compute and assign values dynamically at runtime.

CHAPTER 2

Getting Started

Starting Sybase IQ ETL

1 In Windows, select Start | Programs | Sybase | IQ ETL Development 4.5 | IQ 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 demo projects packaged with the product. Every demo 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 right-click Projects and select New to create a new project.

Setting up a new user account on the demo repository

To add a new user to the demo repository:

1 Select File | Open Repository from the Sybase IQ ETL Development interface.

2 Enter a new client user name in the Client User field.

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

3 Enter a password.

4 Select the Register new user option.

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

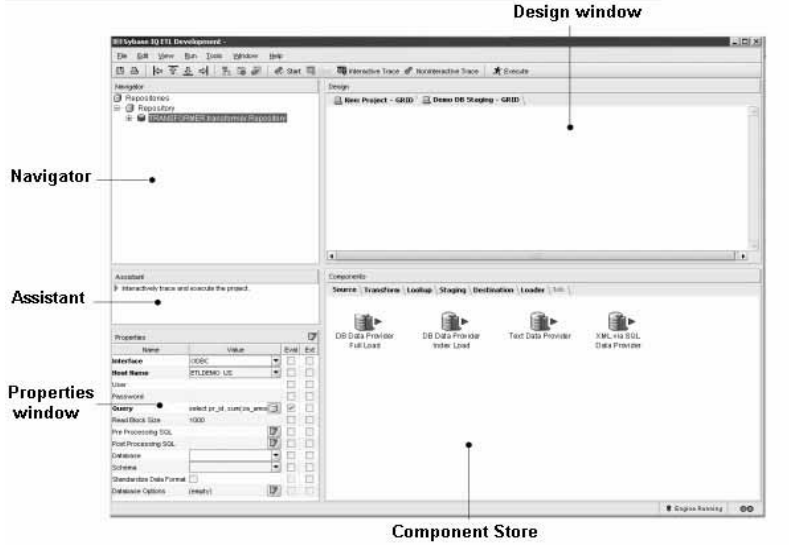
6 Click Logon.

7 Reenter the password and click OK.

Working with the Sybase IQ ETL Development interface

The Sybase IQ 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 the 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 IQ ETL repository is a collection of tables that save and maintain all data related to projects, jobs, and session parameters. You can use Sybase IQ, Microsoft SQL Server, Sybase Adaptive Server® Anywhere (ASA), and Microsoft Access databases, as a Sybase IQ 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, you must 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 client users.

❖ **Opening a repository**

1 Select File | Open Repository. Alternatively, right-click Repositories in the Navigator 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.

In order 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 in the Client User field. The name must be alphanumeric, can contain up to 255 characters, and cannot start with a number.

3 Enter a password.

4 Select the Register New User option.

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

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 17.

❖ **Editing a repository**

1 Select File | Open Repository to open the Repository Logon window.

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 opened simultaneously 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 Right-click a repository name in the Navigator 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 the Show All Objects option 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 selected user is currently connected to the repository, you are prompted to confirm if 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.

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.

Refer to 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 are displayed 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, and can be used to fine-tune and configure the component.

Allowing dynamic expressions

Select the Evaluate option to allow evaluation of dynamic, indirect expressions (SBN expressions). Enter an SBN expression in the corresponding field using the square bracket notation [ ]. 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 selected by default.

❖ **Enabling or disabling evaluation for a property**

1 In the Properties window, right-click the property name that you want to evaluate at runtime.

2 Select Evaluate.

Encrypting properties

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

❖ **Encrypting property values**

1 Right-click a property name in the Properties window.

2 Click 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.

Design window

Use the Design window to:

- Create and modify projects and jobs. See “Projects and Jobs” on page 27.
- Simulate and execute projects. See “Simulating a project” on page 29 and “Executing a project” on page 38.
- Execute jobs. See “Managing jobs” on page 38.

❖ Adding components to the Design window

To create a project or job, you must add and connect components and set their properties. You can add components to the Design window in any of these ways:

1. Select the component in the Component Store and drag it to the Design window. Use this method to add components to projects as well as jobs. The other methods allow you only to add components to projects.
2. Double-click a component in the Component Store.
3. Right-click a component in the Component Store and select Add.
4. 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 input or an output 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. Right-click a component to open the Component pop-up menu. The Component menu displays component-specific commands.

2 Select the action you want to perform.

Component Store

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

Adding components from the Component Store to a project:

- 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 IQ ETL Development:

- Workbench
  - Appearance
  - Data Viewer
  - Query Designer
- Engine

❖ **Customizing preferences**

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

2 On the Preferences window, select Appearance, and set the following options:

- Locale for user interface display – select the locale language for your environment. You can select _de (German), _en_US (US English), or _en_GB (UK English). The default is _en_US.
- Show assistant for creating projects – select this option to view the assistant that guides you through the process of 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 IQ ETL Development for the first time, the default font is Dialog. If you had previously installed Sybase IQ ETL Development, the font is set to the previously defined value. Sybase recommends you to set the font to Dialog.

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

- Create a new project on startup – select this option to automatically create a new project when you start Sybase IQ ETL.
- Create links automatically when components are added – select this option 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 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. The default is Mapping.

- Display qualified transformation objects – select this option 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 – select this option to enforce unique project and job names on a repository connection.
- Show password in component tooltips – select this option if you want the component tooltips to display the password used to log into the underlying database. By default, the password appears in the tooltips as a series of asterisks.
- Use enhanced color accessibility – select this option 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 – select this option 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 – Unselect this option if you want to perform actions without being interrupted by information prompts.
- 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 – select this option to connect automatically to the last logged on repository, on restart.

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 – select this option to delete all records of a selected table when you right-click the table in 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 – select this option to automatically create joins between identical attribute names used within tables or views.
- Use brackets when creating joins – select this option 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 – select this option to start the local engine when you start Sybase IQ ETL.
- Interval between engine monitor updates (sec)– specify the number of seconds to wait between two updates of the Engine Monitor. The default is 5 seconds.
- Rate of simulation (msec) – control the simulation rate by setting Simulation trace delay. 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 – select the option if you want to be able 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 Click Save. For some of the changes to take effect, you may be prompted to restart Sybase IQ ETL. If you do not restart, the changes take effect the next time you start Sybase IQ ETL.

Troubleshooting

During installation, the Sybase IQ ETL installer creates an initial set of data sources. If these repository data sources are lost for any reason, Sybase IQ ETL does not 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 Microsoft Access Driver from the list. Click Finish.

   d In the Data Source Name field, enter DEMO_Repository.

   e Select the *IQETLDEMO_REP.MDB* database from the *Demodata* folder of the installation directory. Click OK.

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

   - Select File | Open Repository to open the Repository Logon window.
   - Select Repository from the Connection list and choose one of the these:
     - Edit
     - Add and enter a name for the connection
   - Select ODBC from the Interface list.
   - Select DEMO_Repository from the Host list.
   - Click Save.

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

   - Driver – Microsoft Access
   - Name – ETLDEMO_DWH; Database – DEMO_DWH.MDB

- Name – ETLDEMO_GER; Database – DEMO_GER.MDB
- Name – ETLDEMO_US; Database – DEMO_US.MDB

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

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

These are the Sybase IQ 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 input and output 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 New | Project from the File menu.

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 Save As from the File menu.

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 gets created, leaving the original untouched, storing no reference 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 Transfer from the File menu.

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

Note Transfer 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 written into the data sinks. 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.

You can simulate your projects in one of these ways:

- Interactively – select Run | Start to initialize the project for simulation, then select Run | Step to manually step through components to simulate project execution.

  Alternatively, select Run | Interactive Trace to initialize the project for simulation and automatically step through components at a predefined pace.

- Noninteractively – select Run | Noninteractive Trace to simulate your project without interaction. See "Simulating projects noninteractively" on page 32.

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

The basic functions of a simulation consist of these high-level steps:

- Start a simulation interactively or noninteractively
- Step through a component
- View the data flow on the connecting link or within the component
- Modify and reinitialize the component to continue to simulate the data flow

In simulation, 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 entire project interactively, select Run | Interactive Trace. If you choose to simulate your project interactively, 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. When you click Start:

- 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.

Note Interactive Trace does not perform postprocessing tasks, such as executing post-SQL statements on DB components.

❖ **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.

Simulating projects noninteractively

To simulate a project without interaction, select Run | Noninteractive Trace. Non Interactive Trace steps through the project and updates connection record counts at the end of the execution.

If the simulation is not already started, selecting this option starts it automatically, and the simulation continues until all data is processed.

Unlike interactive trace, if you simulate your project using non interactive trace, you cannot stop the simulation at any point.

Viewing current mappings

The Mapping Definition window shows the current mapping between attributes of adjacent IN-structures and OUT-structures.

To view current mappings:

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.
- Select 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- and OUT- structures by name, ignoring the specified prefixes.
- Create mapping by Best Match – maps the port attributes of the IN- 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.

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. It should not be a reserved JavaScript keyword. See "Variables" on page 62.

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.

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.

For more information, 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 IQ 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 a simulation is started, 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 is useful while performing a simulation. A large number for Read Block Size can significantly enhance performance when a project executes.

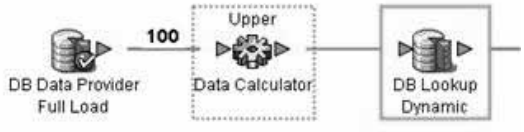
Stepping from current and selected components

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.

Figure 3-1: Current and selected components



The current component is executed next, when you click Step on the toolbar or select Run | Step. During simulation, 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; 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 disabled 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. Sometimes, previewing the content of both the IN-port and OUT-port of a component in parallel tabs is useful.

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. It can also be frustrating to single-step through a large project when you are interested only in simulating a component somewhere in the middle of a complex simulation flow. Select the component and choose Step through or Start through in the Run menu for multi-stepping a project to your point of interest.

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. Most of the Data Sink components offer to customize the Write Block Size. Transformation components such as the Staging component offer to customize values for both reading and writing.

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 down execution when you click Run | Execute. In simulation, the Block Size is restricted to 32K.

Controlling multiple data streams

While most projects consist of a single stream of components connected through links, you can also set up a single project that has multiple unconnected data streams. Since Sybase IQ ETL is a parallel system, 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 design guideline lets you control data streams by connecting the projects to form a job process flow.

Executing a project

You can execute projects in the default grid engine, using one of these ways:

- Select Run | Execute or click the Execute icon on the toolbar. The project currently open in the Design window is executed.

  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 changes in the project 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 selected project is executed.

The Execution Monitor displays. See “Execution Monitor” on page 80.

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 59.

Managing jobs

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

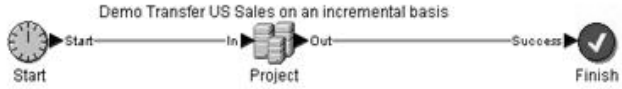
Depending on the success or failure of a project within a job, you can control the job execution.

Job components

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

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

Figure 3-2: Job with the minimum components

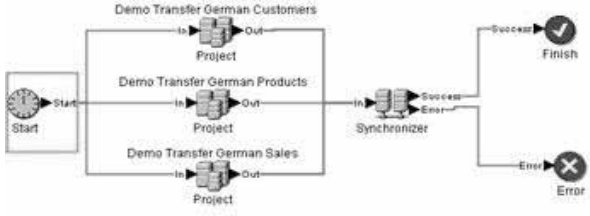


You can extend a job to include multiple:

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

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

Figure 3-3: Job with multiple components



❖ **Creating a job**

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

2 Add 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 IQ 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 Save As from the File menu.

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 reference 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 multi-project 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 Transfer from the File menu.

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 176.

Executing a job

You can execute a job directly from Sybase IQ 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 59.

Scheduling a job

To schedule a job, select Tools | Runtime Manager. Use the Runtime Manager, to create, edit, delete, execute, and terminate tasks. Since Runtime Manager is based on the Windows task scheduling manager, you can find any scheduled ETL task currently defined on the system.

See “Managing jobs and scheduled tasks” on page 59.

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 for migrating data from one database to another.

❖ **Building a migration template**

1. Select File | New | Template. Alternatively, right-click Templates in the Navigator and select this option.
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.
   - Specify the migration type.

     The Migration Type available by default is *DB to IQ*.
   - Select the Allow execution on multiple engines option, to use multiple engines for execution.
   - Click Next.

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

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 the:

- Exclude all objects from transfer icon to exclude all tables.
- Include all objects in transfer icon 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, for more information about database connection parameters you must specify.

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 the Preserve schema/owner option 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 the Use Pipes option is selected, paths are set automatically. If Use Pipes is not selected, manually provide the values ended by the path delimiter. For example, *C:\ETLStage*.

See to “DB Bulk Load Sybase IQ properties list” on page 164 for a detailed description of these properties.

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 to unselect the attributes.

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

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

  Select the Truncate option 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 multi-project 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 perform all tasks described here, also by right-clicking a stored template in the Navigator.

- Save template – If you select this option, the template is stored in the repository. Storing allows you to reuse the collected data for similar jobs.
- Build projects and jobs – Select this option to create one project for each source table, and a migration job that controls the execution of all the projects.
- Create the destination data model – Select this option to 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 – The Execute job option is activated only if the Build projects and jobs option is selected. If you select this option, the generated job is executed after migration template data is processed.

Click Finish.

Note Be sure to select at least the Save template or Build Projects and jobs options if you do not want the collected data to be lost.

Note Before you can execute the generated job, either register engines or open the job and deactivate Multi Engine Execution option. See “Using multiple engines to reduce job execution time” on page 77.

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, double-click the template, or right-click the template you want to modify 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.

2 Select Rename and 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:

1 In the Navigator, right-click the template.

2 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:

1 In the Navigator, right-click the template.

2 Select Create Data Model.

Creating and simulating a sample project

This section describes how to create and simulate a sample project with sample components. For detailed information about components, properties and features, see Chapter 5, “Components.”

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.

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 menu. See “Database connection settings” on page 94 for information about the different Interface types.

2 Select ETLDEMO_US from the Host Name menu.

After 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 Enter a SQL query or click the Query Designer icon to generate the necessary SQL.

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 generated automatically.

8 Click Execute the Query icon to run or test the query. You can also modify 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 menu. See “Database connection settings” on page 94 for information about the different interface types.

2 Select ETLDEMO_DWH from the Host Name menu.

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

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 the Create automatic links when components are added option in the File | Preference window. If the line is not automatically created, click on the output port and drag it onto the input 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 connection point by selecting and deleting the mapping line that is currently linking to it. To delete, select the mapping line and press the Delete key, or right-click and select Delete.

Adding a data calculator

1. Click the Transform tab in the Component Store.
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:

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

   The Data Calculator window has a Tabular and Graph views:

   - Use the Tabular view to enter transformation rules.
   - Use the Graph view to visually define the mapping sequence between the input and output ports.
3. Click the Graph tab. Two sections — IN and OUT 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 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. Without any added function, 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. Click the Start icon on the toolbar to initialize all components.
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. 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.

3 Right-click the link and select Preview to preview data on the link.

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

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 the "Enable delete functionality of database objects" option in the File | Preference window. See "Customizing preferences" on page 21.

- Count the number of records in a table or view.

Opening 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 in the Design window. 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, used to automatically generate select statements that are specific to the records you are working with.
- Navigator – displays all the tables and views under the Model tab, and the recently used tables or views under 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 21.
- Attribute tabs (Select, Join, Where, Sort, Group) and a Generated Query tab – allows 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 IQ ETL Development window.
   b. Select Workbench | Query Designer, then select the Create joins automatically option. See “Customizing preferences” on page 21.
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 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.

Note You can use an asterisk (*) as 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.

Content Explorer

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

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 inspect information about projects and jobs execution and fatal errors, and to review the system log.

1 Select Tools | File Log Inspector. The generated log files are displayed.

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

Note Log files are located in the *log* subdirectory of the installation directory.

- *execution.log* – captures information about job and project execution.
- *fatal.log* – captures low-level information that is written when the system encounters serious unexpected behavior. It includes information from fatal system exceptions when the system can no longer write to the *system* log file.
- *system.log* – captures information about system activities, and operational and exceptional events. The detail level of data written to the *system.log* file depends on the trace level set in the *default.ini* file located in the *etc* directory of the installation folder.

  To set the trace level:

  - Select Tools | Enable System Trace.
  - Use the uTracelevel(*n*) function in a JavaScript procedure.

    The uTracelevel(*n*) function, 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 IQ ETL for the change to take effect.

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

3 (Optional) – Click the 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 A large log file may take some time to load. Until the file is loaded successfully, you cannot perform any other action in the Log File Inspector.

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 create, edit, delete, execute, and terminate tasks.

Since Runtime Manager is based on the Windows task scheduling manager, it includes any scheduled ETL task currently defined on the system.

❖ **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 you want to execute. Click Next.

4 Enter the schedule name. Provide the start time and the end date, and specify how often you want the task to be executed:

- Daily – at a specified time everyday.
- Weekly – at a specified time on particular days of every week. Specify the days of the week.
- 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 Login – runs when you log in.
- At System Startup – runs at system start-up.

Click Next.

5 Enter the user name and password for the Windows user account that executes the schedule. Confirm the password. Click Next.

Note The account information provided must be a valid Windows user account. The user must have read or write access to the ETL user folders such as the installation directory or the Windows user directory, depending on the type of installation.

---

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

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

❖ **Editing a schedule**

1. Select the scheduled project or job you want to edit.
2. Click the Edit a Schedule icon on the toolbar or select Edit from the Actions menu.

❖ **Executing a schedule**

1. Select the scheduled project or job you want to execute.
2. Click the Execute a Schedule icon on the toolbar or select Execute from the Actions menu.

❖ **Deleting a schedule**

1. Select the scheduled project or job you want to delete.
2. Click the Delete a Schedule icon on the toolbar or select Delete from the Actions menu.

❖ **Terminating a schedule**

1. Select the scheduled project or job you want to terminate.
2. Click the Terminate a running Schedule icon on the toolbar or select Terminate from the Actions menu.

After a project or job is scheduled, they are executed under control of the Windows Task Scheduler.

Job execution state codes in the Task Scheduler are shown in the following table.

Table 4-1: Job execution state codes

Result	Type	Description
0	Info	Job or project execution successful
1	Warning	Job or project execution cancelled
101	Warning	No valid license
10001	Error	Cannot retrieve job data from repository
10002	Error	Cannot find initialization component for job
10003	Error	Cannot initialize external job settings
10004	Error	Initialization failed
10005	Error	Job or project execution failed
10101	Error	Connect to repository database failed
10102	Error	No valid repository found in connected database
10103	Error	Cannot open session on connected repository

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 preprocessing and postprocessing 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 IQ 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 Notations (SBN) expressions, which considerably reduces your customization efforts. SBN expressions can be part of component properties (if the Evaluate option is activated for the specific property), SQL statements, or any preprocessing or postprocessing commands. An SBN expression resides inside an opening and a closing square bracket and is evaluated at runtime, as opposed to constant expressions, which are defined at design time.

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, an attribute, or a scalar function. You can use 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')
[uTracelevel(3)]
```

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
- Datatype

The scope of a variable decides in which scope of the environment the variable can be referenced.

The two kinds of variables are:

- 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-Port and OUT-Port. Port variables are valid within the component, and they inherit the name and datatype of the port structure. The name of the variable is either prefixed with IN. for the IN-Ports and OUT. for the OUT-Ports. The 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 high flexibility in transformations, all port and component variables internally use the datatype string. This may result in unexpected behavior is you use numeric values.

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
```

The Port and Component variable names should not be any of the these JavaScript keywords:

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

Reserved JavaScript keywords

package	private	protected	public	short
static	super	synchronized	throws	transient
volatile	const	export		

Functions

Sybase IQ ETL includes a complete set of functions and operators based on a design that supports for Unicode character sets.

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

Square Bracket Notation

Expressions and SQL statements can contain SBN expressions that are evaluated before the expression or SQL statement is executed by the Sybase ETL Server. An SBN expression is surrounded by square brackets [..]. The notation `SBN expression` is used as a synonym for an indirect expression.

You can use SBN expressions in:

- Expressions
- SQL queries
- Pre-SQL and post-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.

SQL statements

SQL queries are used for all components that extract data, mainly the Data Provider and Staging components. Queries are mandatory for those components because they define OUT-port structure.

To enter a query for the component, click the Query icon in the Properties window. You can:

- Enter a query.
- Run a query.
- Save a query.
- Open Query Designer. See “Query Designer” on page 53.
- Look up the database schema.

Entering queries

1 Enter a select statement in the Query field. Alternatively, use Query Designer to create ad hoc queries. See “Query Designer” on page 53. You can select the tables and attributes from the available database schema for your query. See Looking up the database schema.

2 Click the Execute Query icon.

Note After you change an existing select statement, initialize the component before executing another step.

Validating queries

The query is immediately validated against the database system connected to the component. Therefore, the query syntax must be compliant with the native SQL dialect the connected database system is using. Using SQL92 ANSI Standard queries allows you to switch to different database systems without changing the select statements.

Looking up the database schema

You can select tables and attributes for your query from an available database schema.

1 Click the Query icon in the Properties window.

2 Click the Database Lookup icon.

3 Select the tables and attributes you want to use in the query and click OK.

Using SBN expressions in queries

The following examples show how to use SBN expressions in queries.

Examples

A select statement to retrieve a specific customer record might include a constant customer record CU_NO 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 IQ ETL functions inside the SBN expression. The following statement returns the same record using a value of *“1234”* for CustNo1 and a value *“4567”* for CustNo2:

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

Preprocessing and postprocessing SQL statements

For any component with database connectivity, you can enter preprocessing and postprocessing SQL statements in the Property window of the components. Those properties allow you to enter SQL statements that are executed during initialization (preprocessing) of the component or after completion (postprocessing) of the project.

Some considerations:

- The SQL statements do not return output after execution.
- Any SQL statement accepted by the connected database system is allowed.
- You can enter multiple SQL statements in the preprocessing or postprocessing SQL property by using a semicolon; as a statement delimiter.
- SBN expressions are allowed in preprocessing and postprocessing SQL.

These examples show preprocessing and postprocessing SQL:

```
delete from products;
update customers
set cu_desc = 'valid';
```

Using the JavaScript Editor and Debugger

JavaScript is an object-oriented scripting language designed for embedding 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. JavaScript Editor and Debugger let you interactively edit, debug, and execute JavaScript code.

Features

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 the following features:

- Color-coded syntax for better readability
- Watchlist to control the assigned values of variables and attributes when running or stepping through the code

- Multiple user-defined breakpoints to stop code execution at any line positions
- User-defined Go points to arbitrarily choose the position from which a code shall execute
- Step mode to execute the 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 input and output 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 and Output Record, 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.

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 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. At any time, you can check the current value of a variable.

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

❖ Stepping through the code

Note The JavaScript Editor and Debugger works without input data at the input port of the component. However, for best results, populate the input 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

Rather than stepping through the procedure line-by-line, include Breakpoints at selected lines.

1 To include or remove breakpoints, click the line where you want to set the 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.

Inline inspection of variables

You can perform inline inspection of the current value of a variable when stepping through the code in Debug mode or after the code, is executed. Right-click the variable to view the variable name and value.

Monitoring values in the watch list

You can use the watch list to monitor the changes of variable values during the execution of the code. 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

Interrupting execution

From inside the Editor, click the Cancel a running script icon on the toolbar, to interrupt JavaScript execution.

Creating user-defined errors

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

```
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 IQ ETL are represented as strings. This may result in unexpected behavior when working with numeric values. 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

Use the uScriptLoad("*filename*") function to include external files into a script. 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. Right-click the component in the Design window 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. Right-click the component in the Design window 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 an expression 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 section Special property values.

❖ **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. The Parameter Set window opens with an additional Execute button.

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 80.

Note If no stored parameter set is available, the window automatically opens in edit mode. See “Managing jobs and scheduled tasks” on page 59 for additional information about assigning parameter sets to scheduled projects and jobs.

Assigning parameter values

Selecting parameters

- 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.

To enter or edit parameter values

After you select a parameter:

- Enter the new value. The old one will be overwritten.
- Edit the existing value directly in the Value cell.
- Choose Edit from the popup menu on the Value cell and edit the value. 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 to use the value for all selected lines.

Alternatively, click the Edit Selected values button or choose Edit selected from either the Edit or popup menu, edit, and 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 the values displayed on project design differ from the values that you must provide in a parameter set.

Check boxes

For properties represented by a checkbox on project design, you must provide 0 (deactivated) or 1 (activated) as the value.

Expressions

To use variable values in a parameter set, enter 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 64.

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

Note Expressions cannot be validated 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

Component	Property	Prompt	Value	Evaluate
	Column Delimiter	Position		
		Tab	[uChr(9)]	x
		Comma	,	
		Semicolon	;	
	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: A user defined name for the engine.
- Host: The name or IP address of the engine host.
- Port: The number of the port the engine is listening on.

- Base Rank: A user defined ranking for the engines. A job first tries to executes the projects on the highest ranked engines.
- Description: A description for the server.

You can register individual or multiple ETL servers manually.

❖ **Manually registering a** grid **engine**

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

2 Enter the desired 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 multi-engine jobs

You can run a job on multiple engines using the parallel GRID architecture. A typical multi-engine job contains multiple projects with no or little dependencies between them. Thus, the 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, schedule it to run in the Runtime Manager.

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 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 detailed information about all the engines. You can specify the number of seconds to wait between two updates in the Update Interval field. The default value is 5 seconds.

Execution Monitor

Execution Monitor displays the properties of the current job or project. To display the Execution Monitor:

1 Select the project you want to execute.

2 Click Execute on the toolbar.

The Execution Monitor window is divided into two panels, Job and Projects.

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

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 displayed 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 IQ 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**

1 **Viewing performace data overview of a selected project or job**

Right-click a project or job in the Navigator 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 only one project or job at a time.

To 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! This will permanently remove the selected execution performance data from the performance log. This data cannot be recovered after deletion.

---

2 **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 appears displaying details, such as component name, duration, records read, and records written.

To view performance details for a selected component, select a row in the tab displaying the componet 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 displayed. The component performance details such as, event name, duration, records read, and records written are displayed.

Note You cannot view the details of the Load Project component. This component represents the time it takes the engine to load the project before execution.

To go back 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.

3 **Viewing job performance data**

To view job 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 appears displaying details, such as project name, duration, records read, and records written.

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.

Note You cannot view the details of the Load Project component. This component represents the time it takes the engine to load the project before execution.

To go back 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.

Resetting performance data queries – The performance data queries are stored in the system registry under *HKEY_CURRENT_USER\Software\JavaSoft\Prefs\sybase\sybetl*. If you experience errors while using Performance Data, you need to reset these queries to their default values. To reset the modified queries to the default value at any point, delete the Performance Data Query keys in the registry, and restart the application.

Warning! Do not to change the queries unless absolutely necessary.

❖ **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.

The searched data is highlighted in the table.

❖ **Printing performance data**

1 Select Tools | Generate Report.

2 Select the level of details that you want to print and choose the destination file. The available options are:

- Overview
- Job Performance (displayed only when 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 the Only selected executions checkbox.

Note The Only selected executions checkbox appears disabled 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 both tabular as well as graphical data. It includes a table of contents that provides links to corresponding sections within the report. Each report can include the following sections, depending on the level of details you have selected to print:

- Overview
- Job Performance (displayed only in a job performance data report)
- Project Performance
- Component Performance

The Performance Overview section 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

The performance data is stored in a single, de-normalized, repository table named TRON_PERFORMANCE, which you can query to collect performance data. This section describes the information included.

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 have a result (like succeeded or failed). In addition you will find information about the engine that reported an event.

The following list gives a short description of the reported events:

Class	Name	Description
control	execute job	Total execution time of a job (1 record per job execution, duration in attribute PRF_JOB_DURATION (Job Duration in ms)).
init	load job	Get job definition from repository.
control	execute project	Total execution time of a project (1 record per project execution, duration in attribute PRF_PRJ_DURATION [Project Duration in ms]).
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 pre-processing.
perform	process	Perform component step.
perform	finish	Perform component post-processing.
perform	read	Get data to component input port.
perform	write	Push data from component output port.
finish	close	Close project and component instances.

Note Due to distributed, multi-threading the total project execution time can be significantly lower than the sum of the execution time of all participating components.

General information

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 the project or job is located in.

Job execution information

For a job you will find the ID, the version (modification date), and the name. A single job execution event will store the duration of a job (in minutes). The components (projects) of a job are represented by their ID, class, and version.

Project execution information (including job projects)

For each executed project ID, version (modification date), name, and a global, unique execution ID is provided. A single project execution event will store the duration of a project (in minutes).

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

Port events provide the ID, name, class, type and the amount of input or output blocks and records.

CHAPTER 5

Components

This chapter provides detailed descriptions of the various Sybase IQ ETL components.

Overview

Sybase IQ 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 has to start with one or more Source components. Source components have no IN-ports and one or more OUT-ports.
- Transformation components, Lookup components, and Staging components – apply specific transformations to the data in the transformation stream. These types of components have at least one IN-port and one OUT-port.
- 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.

Note While different from a functional point of view, all components share common concepts.

Setting up component properties

Each component is dedicated to a specific task and incorporates task-specific features. However, all components follow the same procedure to set up their properties.

Before you can use a component within a project, you must set the required properties. The properties can be set in:

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

Evaluating SBN expressions

You can allow Square Bracket Notation (SBN) expressions that are evaluated before using the property value, using the Evaluate property. See “Square Bracket Notation” on page 64.

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

❖ **Changing evaluate properties**

You can change the evaluate properties in one of the following ways:

1 In the Properties window, select the Eval option for the property.

2 Right-click 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 repository. However, most entries in the Sybase IQ ETL repository are not encrypted, but are represented as readable character sets.

To toggle encryption properties, select the Encrypt checkbox, or right-click the property in the Properties window, 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. Like other properties, these incorporate variables that you can reference in component expressions or procedures. They can contain expressions that are evaluated at runtime or be assigned via parameter sets.

❖ **Adding custom properties**

1 Select the component in the Design window.

2 Right-click in the Property window and select Add.

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

4 Enter a value in the prompt field. This is the label that appears in the Properties window.

5 Enter a value in the description field. The description appears in the property tooltip.

6 Click OK.

❖ **Removing custom properties**

1 In the Property window, select the custom property you want to remove.

2 Right-click and select Remove. Click OK to confirm.

Note Make sure to 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 Right-click the component in the Design window 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 stepping 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 additional 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 – port structure is defined and one or more mandatory properties are not defined.

- Red – no port structure is defined and one or more mandatory properties are not defined.

Note You can change the color of the component ports to enhance support for users with color disabilities by selecting the Use enhanced color accessibility option in the File | Preference window. See “Customizing preferences” on page 21.

Modifying port structures

❖ **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 For port structures that 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. It should not be a reserved JavaScript keyword. See “Variables” on page 62.

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.

❖ **Selecting a port that serves 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 is displayed in the lower area of the window.

2 Click Apply.

Simulating components

Initializing components

1 Select the component in the Design window.

2 Right-click and select Initialize, or Initialize and Step.

Note 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 steps 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 the component repetitively, the same set of records at the IN-port are reprocessed in each step and forwarded to the output port without increasing the number of the record set at the OUT-port. For many components, the stepping can be done from inside the component window or by using the menu that appears when you right-click in the Sybase IQ 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. A data block contains a subset of records. The number of records in each subset is related to the Read Block Size parameter displayed 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 the 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

You can specify database connection parameters in the:

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

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

  - Sybase
  - Sybase 15
  - SQLite Persistent

    Note Use SQLite Persistent for your test environment only. Do not use it in a production environment.

  - IBM DB2
  - ODBC

    Note The ODBC driver must be installed on the same computer as Sybase IQ 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.

  - Oracle

- OLE DB

Note Sybase Open Client™ must be installed on the same computer as Sybase IQ 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 the project is to be executed on an ETL Server, the ETL Server must also have access to Open Client libraries.

- 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 270.

- 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/owner to restrict the objects displayed and create new tables in that schema.
- Standardize Data Formats – select this option to convert incoming date and number information into a standard format, which Sybase IQ ETL can move between systems that support different formats.
- Database options – Select this options to override performance defaults and control the behavior of some transactions. For a list of database options, see "Interface specific database options" on page 263.

Source components

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

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)
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.

DB Data Provider Full Load

DB Data Provider Full Load is a Source component that extracts data from any data source accessible by an ODBC connection or native driver (DB2, Oracle, Sybase). The structure of the single output 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, and click the Properties icon in the Properties window.

2 Enter the connection parameters. See “DataProvider Full Load properties list” on page 97.

Note 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 simple query directly in the Query window, or click the Query Designer icon to open Query Designer and generate queries. See “Query Designer” on page 53.

4 Click Finish.

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

❖ **Updating port structure**

To update the port structure with changes made to the database:

1 Right-click the DB DataProvider Full Load component.

2 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.

DataProvider Full Load properties list

DataProvider Full Load properties list identifies the connection parameters and other items you need to define in the Property window. The properties that are mandatory appear in **bold** text.

Required properties

Property	Description
Interface	Identifies the method or driver to use to connect to the data source.
Host Name	Identifies the data source. Options available on the Host Name list depend on the interface you select.
Query	Click the Query icon to open a window where you can create a query that retrieves information from the data source. You can use the Query window to create a simple query, or click the Query Designer icon to open Query Designer and generate a query. See “Query Designer” on page 53.

Optional properties

Property	Description
User and Password	Used in combination to identify an authorized database user, and to protect the database against unauthorized access.
Read Block Size	Determines the number of records retrieved by the component in a single step.

Property	Description
Pre-processing SQL	Click the Pre-Processing SQL icon to open a window where you can create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-Processing SQL icon to open a window where you can create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identifies the database you want to use as data source. If you select this option, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Converts incoming date and number information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to open a window where you can set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

Data Provider Full Load Demos

Sybase IQ ETL includes several demonstrations for the Data Provider Full Load component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Source*, select *DB Data Provider - Full Load*.

DemoRepository

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, and click the Properties icon in the Properties window.

2 Add the connection parameters. See the “DataProvider Index Load properties list” on page 100 for specific field requirements.

Note You must add a valid interface and host name.

3 Click Finish.

4 In the Properties window, click the Ascending Index icon and 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 53.

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:

1 Right-click the project in the Navigator.

2 Select Reset Execution Properties.

❖ **Updating port structure**

To update the port structure with changes made to the database:

1 Right-click the DB DataProvider Index Load component.

2 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.

DataProvider Index Load properties list

DB Data Provider Index Load properties list identifies the connection parameters and other items you define in the Database Configuration window. The properties that are mandatory appear in **bold** text.

Required properties

Property	Description
Interface	Identifies the method or driver you want to use to connect to the data source.
Host Name	Identifies the data source. Options available on the Host Name list depend on the interface you select. If you use SQLite as a host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.

Query	Click the Query icon to open a window where you can create a query that retrieves information from the data source. The selection criteria in the WHERE clause needs to be qualified using the predefined variable *LoadIndex*. Enclose the 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. On Microsoft Access databases, use # for datetime values.
	You can use the Query window to create a simple query, or click the Query Designer icon. See "Query Designer" on page 53.

Ascending Index	Opens a window where you can select the attribute that contains the ascending index for delta load.

Optional properties

Property	Description
User and Password	Used in combination to identify an authorized database user, and to protect the database against unauthorized access.

Property	Description

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

Read Block Size	Determines the number of records retrieved by the component in a single step.
Pre-processing SQL	Click the Pre-processing SQL icon to open a window where you can create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to open a window where you can create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Identifies the database you want to use as data source. If you select this option, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Converts incoming date and number information into a standard format that Sybase IQ 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 with a '`.`' as the decimal separator.

Property	Description
Database Options	Click the Database Options icon to open a window where you can set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

Simulating an Index Load

When your project containing an Index Load component is fully configured you can simulate the incremental load by performing these steps:

1 Select Run | Noninteractive Trace to run the project noninteractively. All records matching the condition specified by the Load Index Value property are processed.

2 Simulate the project interactively. Index Load component will not return any records. See “Simulating a project” on page 29.

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

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

DataProvider Index Load Demos

Sybase IQ ETL includes several demonstrations for the Data Provider Index Load component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Source*, select *DB Data Provider – Index Load*.

DemoRepository

In 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 component window, select the text file you want to use as a data source.

3 In the Properties window, modify the file description properties, if necessary.

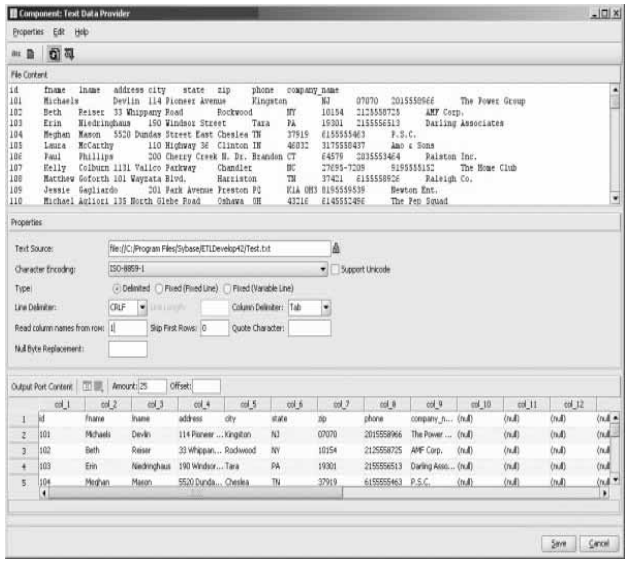
See “Text Data Provider properties list” on page 105 for specific field requirements.

Text Data Provider component window

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 – displays the file description properties.

- Output Port Content pane – displays a tabular view of data at the OUT-port.



Text Data Provider properties list

Text Data Provider properties list identifies items about the structure of source file. Properties are initially set when you add the component to the project. The properties that are mandatory appear in **bold** text.

Required properties

Property	Description
Text Source	Identifies the text file you want 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	Click the Columns icon to open a Property sheet that lets you define columns for the data in your source file.

Optional properties

Property	Description
Row Delimiter	Select 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	Select how columns are delimited: • Position (fixed column positions) • Tab • Comma • Semicolon Alternatively, you can enter a different delimiter.
Column Quote	Specify how the values in the source file are quoted: • None • Single quote • Double quote
Fixed by Bytes	Specify how to interpret the values provided for line length, column start, and column end: • Not selected – the values are interpreted as number of characters. This is the default. • Selected – the values are interpreted as number of bytes.
Null Byte Substitute	Sets the character to replace null bytes.
Skip Rows	Skips a specified number of rows in the row sequence.
Encoding	Sets the current character encoding. To set the character encoding, select the appropriate value from the drop-down menu.
Support Unicode	Sets Unicode support.
Read Block Size	Determines the number of records retrieved by the component in a single step.

Text Data Provider Demos

Sybase IQ ETL includes several demonstrations for the Text Data Provider component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Source*, select *Text Data Provider*.

DemoRepository

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

XML via SQL Data Provider 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 output port.

XML via SQL Data includes one output port by default, but you can add ports. Any OUT-ports you add in XML Port Manager appear in the Design window. For additional information, see “Working with XML Port Manager” on page 108.

❖ **Updating the output port structure**

To update the output port structure to reflect changes made to the XML source file:

1 Right-click the XML via SQL Data Provider component.

2 Select Reconfigure.

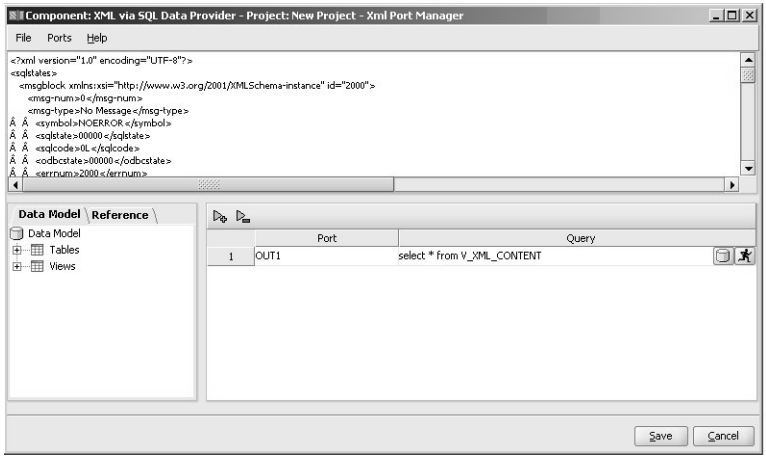
The Reconfigure option updates the component configuration when there is a change in the database schema. It closes the current connection, opens a new connection to the database, reads the XML source file, and applies the updates to the output port structure.

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.

- OUT-Port area – used to write queries against the data model, and send the results to a particular OUT-port. Although XML Port Manager is configured with one OUT-port by default, 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. 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 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.

  You can add an Info Port to forward the XML document to the next component. This port is visible after you exit XML Port Manager.

Setting up a sample project

This section guides you through setting up the 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 IQ 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 have to 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. The properties that are mandatory appear in **bold** text.

Required properties

Property	Description
XML Source	Identifies 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	Click the Data Output icon to open XML port manager. XML port manager is a management console where you can query the XML source. See “Working with XML Port Manager” on page 108 for more information.

Optional properties

Property	Description
Document Schema	Click the Document Schema icon to identify an external schema (*.xsd*) or DTD that you can use to validate the XML source.
Namespace Schema	Points 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.

Property	Description
Validate Schema	Select this option if you want to enable schema and DTD validation.
XML Options	Click XML Options to open a window where you can set these XML processing options: • Process namespace - Set this value to 0 if you do not want namespace specification to be considered during parsing. The default value is 1. • Full schema check - Set this item to 1 to check for items that may be time-consuming or memory intensive. The 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.
DB Schema	Click the DB Schema icon to select the database schema setup (create tables) script. Use this option to enforce a fixed data model.
DB Schema Options	Click the DB Schema Options icon to open a window where you can customize the settings for tables and attributes generated from the XML structure, including the prefixes for table and attribute names.
Database Options	Click the Database Options icon to open a window where you can set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.
Read Block Size	Determines the number of records retrieved by the component in a single step. If the component has more than one output 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 IQ ETL includes several demonstrations for the XML via SQL Data Provider component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Source*, select *XML via SQL - Data Provider*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer.Repository* | *Projects*. Then select Demo XML via SQL Data Provider.

Transformation components

Transformation components have at least one IN-port and one OUT-port and apply specific transformations to the data in the transformation stream.

Component	Description
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.
Character Mapper	Replaces characters and strings in an input record. Character Mapper applies replacement mapping to all selected attributes.

Data Calculator JavaScript

Data Calculator JavaScript is a Transformation component that lets you define rules that apply transformations on 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 on the attribute level. You provide lookup data at special lookup ports.

Impact on simulation sequence:

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 Link the IN-port of Data Calculator to the OUT-port of the component that provides the inbound data, if necessary.

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 click the Rule icon in the Properties window. The component window opens and prompts you to assign a default mapping by order.

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 map attributes in the IN-port to a corresponding attribute in the OUT-port individually. See “Mapping port attributes” on page 118.

For more information about mapping options and transformation rules, see “Working with the component window” on page 116

Working with the component window

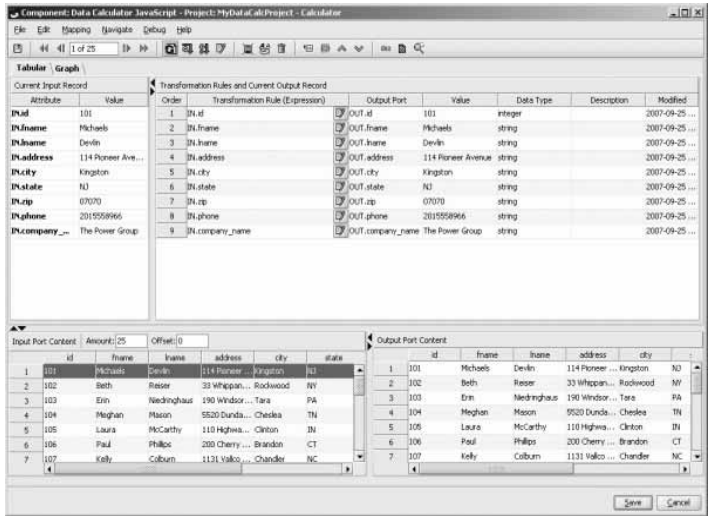
The Data Calculator component window provides tabular and graphic views of the data stream. You can also use this window to map port attributes and define transformation rules.

Tabular view

The tabular view provides a structural view of the current record structure, port attributes, and transformation rules. It includes:

- The 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.
- The 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. OUT-port values reflect IN-port values by default. Changing the attribute value or transformation rule changes the OUT-port values.

  If the transformation rule is a functional expression, you must enter it in 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 need to set the OUT-port attribute value.
- The Input Port Content area shows the current set of records available at the IN-port.
- The Output Port Content area shows the current set of records available at the OUT-port.

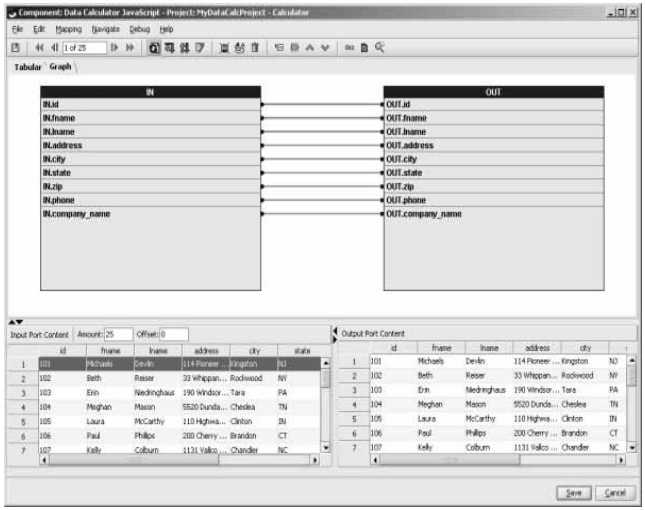


Graph view

The graph view shows the current mapping between the IN-port and OUT-port attributes.

- The Input Port Content area shows the current set of records available at the IN-port.
- The Output Port Content area 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 Data Calculator 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 structures in one of these ways:

- Select Mapping in the menu bar, and select one of these predefined mapping sequences:

- Create mapping by Order – maps the port attributes of the IN- structures and OUT- structures sequentially.

  Note If the number of attributes is different on both sides, some of the port attributes are not mapped.

- Create mapping by Name – maps the 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.

- 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

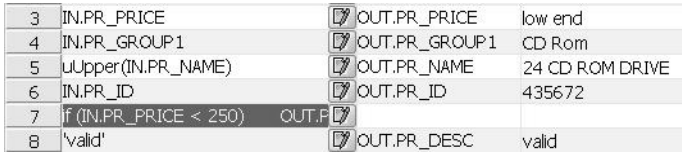
One of the essential features of Data Calculator is the capability to display the result of transformation rules immediately. This simulation capability allows you to verify incoming data, test transformation rules, and view the effect of the rules on data output.

Current input record attribute values

By default, output port values reflect Input port values. Changing an Input port attribute value lets you test transformation rules, and to see the results in the current output record. Manual modifications made in this area affect only the data of the Input/Output port buffer. This offers a convenient way to create test cases for transformation rules.

Editing transformation rules

In the Transformation Rule column, you can add, modify, or delete transformation rules. You can also edit single-line functions by changing the current attribute input field. Adding the function uUpper to the attribute IN.PR_NAME looks like the following example:



You can use the Procedure Editor. See “Using the JavaScript Editor and Debugger” on page 67 for information about creating complex procedural transformations.

You can also add transformation rules. This is extremely helpful if the number of OUT attributes does not match the number of IN attributes, or if you add additional attributes to the project at a later stage of development.

Note The graphic view mirrors the actual port structures. You cannot add or delete attributes there.

❖ **Managing transformation rules**

1 To add a transformation rule, right-click anywhere on the Transformation Rule or Current Output Port column.

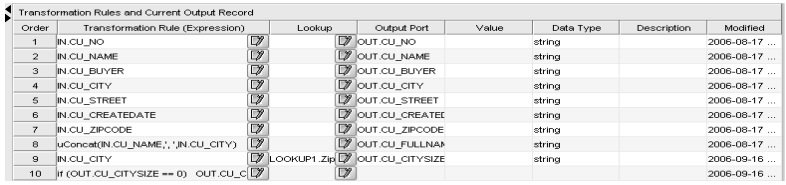
2 Select Insert.

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.

Sequence of processing transformation rules

Transformation rules are processed in sequential order. The processing starts with the first transformation rule of the list. Consider the following example:



Transformation Rules and Current Output Record

Order	Transformation Rule (Expression)	Lookup	Output Port	Value	Data Type	Description	Modified
1	IN.CU_NO		OUT.CU_NO		string		2006-08-17 ...
2	IN.CU_NAME		OUT.CU_NAME		string		2006-08-17 ...
3	IN.CU_BUYER		OUT.CU_BUYER		string		2006-08-17 ...
4	IN.CU_CITY		OUT.CU_CITY		string		2006-08-17 ...
5	IN.CU_STREET		OUT.CU_STREET		string		2006-08-17 ...
6	IN.CU_CREATEDATE		OUT.CU_CREATED		string		2006-08-17 ...
7	IN.CU_ZIPCODE		OUT.CU_ZIPCODE		string		2006-08-17 ...
8	uConcat(IN.CU_NAME,', ',IN.CU_CITY)		OUT.CU_FULLNAM		string		2006-08-17 ...
9	IN.CU_CITY	LOOKUP1.Zip	OUT.CU_CITYSIZE		string		2006-09-16 ...
10	if (OUT.CU_CITYSIZE == 0) OUT.CU_C						2006-09-16 ...

Line 9 looks up the number of zip codes for the city and stores the result in CitySize. The procedure in Line 10 calculates the CitySize string based on the CitySize number.

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 the outgoing data. Depending on 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:

Autosynchronization 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 manually apply all transformation rules to all current records at the IN-port, click the Step icon on the toolbar.

Fetching another set of records:

To fetch another set of records, click the Step through the next incoming data buffer icon on the toolbar.

Stepping through input port records:

To step through records at the IN-port:

- Click the appropriate record control on the toolbar.
- Click Navigate and select the appropriate option.

- Select a record from the Input Port Content list.

Note The values shown on the Current Input Record area are updated as you change the current record.

Search for keywords in the transformation rules:

Click Search the Content of Transformation Rules icon on the toolbar.

Highlight null and empty values

Click Highlight NULL-Values and Empty Values icon on the toolbar.

Using lookups in Data Calculator

Data Calculator performs lookups on the attribute level. You must enter the 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 created and connected automatically. Alternatively, right-click the Data Calculator and select Add Input Port. Connect the OUT-port of the data providing component with the new port. The number of lookup ports is unlimited.

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.

If you want to look up compound keys, you must 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 its current option values.

- Lookup Name — is inherited from the associated port and cannot be changed here. To change a name of a port, select Description from the port menu.

- Lookup Size — enter the estimated number of lookup records to optimize memory allocation and lookup performance. See “DB Lookup” on page 132.
- 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 open the Tabular tab in the Data Calculator window. If Lookup ports are available, an additional Lookup column appears.

Provide the following information for each lookup rule:

- Key Expression – This is the value to search for in the first column of the lookup list. You enter the key expression (for example IN.PR_ID) as the 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. To specify the value to return, 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 adding or removing attributes.

- Output Variable – the Lookup return value is assigned to this variable. You can select a variable from the pop-up menu on the Output Port column (for example, OUT>>OUT.PR_NAME).
- Default Expression (optional) – 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, under the transformation Rules and Current Output Record panel. The Lookup Options window is used to specify a default expression. It includes popup menus that allow you to select Lookup and Output Attributes.

Data Calculator JavaScript Demos

Sybase IQ ETL includes several demonstrations for the Data Calculator component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Transform*, select *Data Calculator - JavaScript*.

DemoRepository

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

Data Splitter JavaScript 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 Data Splitter's IN-port to the OUT-port of the component that provides the inbound data.
3. Link Data Splitter's OUT-ports to the IN-ports of the components where you want to direct the outbound data.
   - OUT1 is the top OUT-port.

- OUT2 is the bottom OUT-port.

You must configure the IN-port structure of the components where you want to direct the outbound data.

4 Double-click the Data Splitter JavaScript component.

5 Add the conditions you want to use to direct the data flow:

a Right-click the port (OUT1, OUT2) you want to add the conditions to and select Edit.

b On the Condition window, define the conditions for each column that you want to apply.

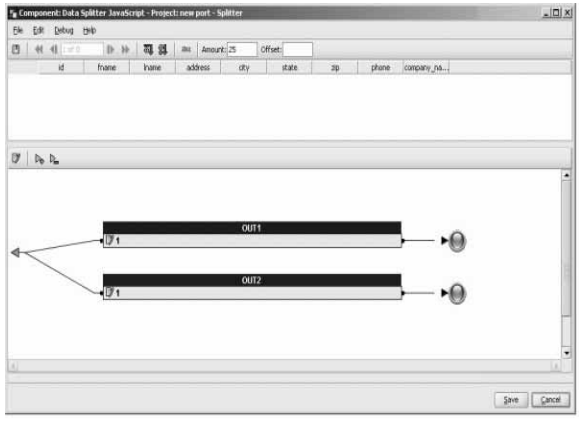
c Click Save.

Splitting inbound data

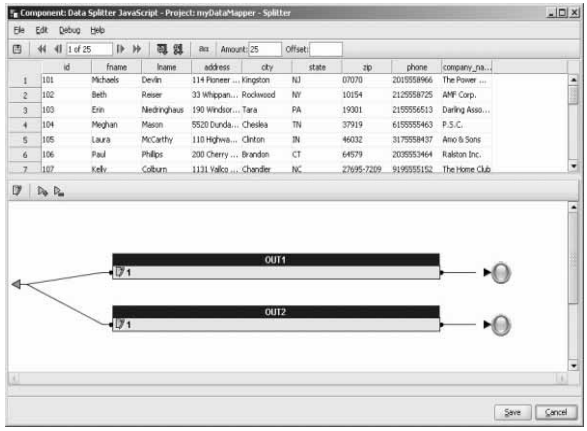
Adding Data Splitter to a project opens a component window that displays inbound data attributes and OUT-port conditions.

Data Splitter is configured with two OUT-ports, OUT1 and OUT2. The OUT1 port is the top OUT-port; The OUT2 port is the bottom OUT-port.
IN-port conditions for OUT1 and OUT2 port are pre-set to 1. Since this condition is always true, regardless of the current values if the IN-port, Data Splitter will copy all incoming records to both the output ports.

Since the IN-port data buffer is initially empty, only the inbound data attributes are visible. The OUT1 and OUT2 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.



There is a relationship between the records in the Input area and the OUT-port. Since the OUT1 and OUT2 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.

Note The number of records that Data Splitter forwards to the OUT-ports can be different from the number of incoming records. If a single record matches more than one port condition, it will be available on all of these ports. Records that do not match any of the conditions are removed from the data stream.

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 Data Splitter evaluates the condition, the result is either TRUE (1) or FALSE (0).

You may, for example, want to write one set of conditions for OUT1 and another set of conditions for OUT2.

❖ Modifying port conditions

1 Double-click the Data Splitter JavaScript component to open the Data Splitter component window. Right-click the port and select Edit.

2 On the Condition window, 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 from the IN pop-up menu.

❖ **Adding new output ports**

Although Data Splitter 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 output port. Click OK.

❖ **Removing output 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.

Note You cannot delete all ports. The component must have at least one output port.

Special port conditions

Port conditions determine how Data Splitter distributes records. You can define overlapping port conditions that are not mutually exclusive.

- 1 — TRUE. All records are forwarded to this port, including records that concurrently match with any other port condition.
- (empty) — A port with an empty condition collects all records that did not match any other condition defined in the Data Splitter.

Data Splitter JavaScript Demos

Sybase IQ ETL includes several demonstrations for the Data Splitter component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Transform*, select *Data Splitter - JavaScript.*

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer.Repository* | *Projects*. Then select:

- Demo Data Splitter
- Demo Text Data Sink Delimited/Fixed

Character Mapper

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, 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 OUT-port of the Character Mapper to the IN-port of the component where you want to direct the outbound data.

   You will need to 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 button on the toolbar to populate the Input and Output content.

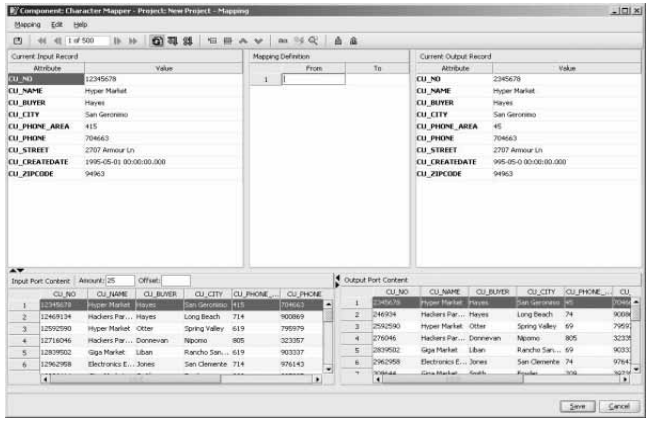
4 Add a mapping definition. See “Creating new mapping definitions” on page 129.

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, which displays the columns, rows, and content for the record currently at the IN-port.
- The Current Output Record pane, which displays the columns, rows, and content for the current record as it appears to the OUT-port.
- The Mapping definition pane, which 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. 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. The selected mapping definition is deleted.
- To change the order of the mapping definitions, select the Move row up and Move row down icons on the toolbar.
- 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. If you want to exclude columns from a character mapping, click Exclude in the Properties window, and select the columns you want to exclude.

❖ **Reusing mapping definitions**

Character mapping definitions can be saved to a file, allowing to reuse them in other projects. To save the character mappings to a definition file, and reuse them in other projects:

- Click Export character mapping and Import character mapping icons on the toolbar. See Exporting mapping definitions and Importing mapping definitions for more information.

❖ **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.

Note 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 IQ ETL includes several demonstrations for the Character Mapper component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Transform*, select *Character Mapper*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer.Repository* | *Projects*. Then select Demo Character Mapper.

Lookup components

A Lookup operation looks up a value corresponding to a key in a lookup table containing a list of key and value pairs. A static lookup table can be cached during the execution of a project, or the 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 returning 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 will be 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	Uses a query that references the key value in the query's `WHERE` clause to perform a dynamic lookup. 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 might be modified by concurrent database users (or even within the same project). In this case a non-dynamic database lookup might search for invalid data. Use DB Lookup Dynamic to ensure you locate the current value.

DB Lookup

DB Lookup looks up values in a database. The lookup data is specified by the result set of a query returning 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 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 133 for specific field requirements.

4 In the Properties window, specify the Key Attribute and Value Attribute. These are the Lookup values.

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 United Stated of America. 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

DB Lookup properties list identifies the connection parameters and other properties you define on the Database Configuration window. Required properties are labeled in **bold**.

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 might refer to the same attribute of the record structure, thus allowing the overwrite of a key with its corresponding value.

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Specify the data source. Options available on the Host Name list depend on the interface you select.
Query	Click the Query icon to create a query that retrieves information from the data source. You can use the Query window to create a simple query, or click the Query Designer icon to open the Query Designer.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.
Default Value	Specify a Default Value to assign to the value attribute. DB Lookup uses this value when it cannot find the key value in the lookup table.
Use Key Value	Select the option to assign the key value to the value attribute instead of the default, if the lookup fails.
Lookup Empty/Null Keys	Select the option to performs 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	Specify the database you want to use as data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Select the option to convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.

Property	Description
Database Options	Click the Database Options icon to set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Lookup Demos

Sybase IQ ETL includes several demonstrations for the DB Lookup component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Lookup*, select *DB Lookup*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer.Repository* | *Projects*. Then select:

- Demo DB Lookup
- Demo Transfer German Products

DB Lookup Dynamic

DB Lookup Dynamic uses a query that references the key value in the query’s WHERE clause to perform a dynamic lookup. 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 might be modified by concurrent database users (or even within the same project). In cases like this, a non-dynamic database lookup might search for invalid data. Use DB Lookup Dynamic to ensure you locate the current value.

Another typical use case for this component is a lookup table that exceeds the memory available on the local machine. By using the DB Lookup Dynamic component the lookup is slower, 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 137 for specific field requirements.
4. Specify the Key Attribute and Value Attribute. These are the Lookup values.
5. 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 value from the source table.

❖ **Resetting the Lookup key value**

To reset the value of the Key Attribute and Value attribute properties of the component to default:

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 at DB Lookup Dynamic component therefore contains those three attributes.

The table to lookup 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 will be replaced by the string `"INVALID"`. A successful lookup will replace the German product number by 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 are displayed in **bold** text.

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 will overwrite 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	Specify the data source. Options available on the Host Name list depend on the Interface you select. If you use SQLite as a Host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.
Query	Click the Query icon to create a query that retrieves information from the data source. You can use the Query window to create a simple query, or click the Query Designer icon to open Query Designer.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.

Property	Description
Default Value	Specify a Default Value to assign to the value attribute, in case the key value was not found in the lookup table.
Use Key Value	Select the option to assign the key value to the value attribute instead of the default, if the lookup fails
Lookup Empty/Null Keys	Select the option to 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.
Database	Specify the database you want to use as the data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Select the option to convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to 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 IQ ETL includes several demonstrations for the DB Lookup Dynamic component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Lookup*, select *DB Lookup – Dynamic*.

DemoRepository

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 output 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 given size.

You can use Staging components to perform sorts or aggregations on formerly unordered records 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 could also use this component for creating an intermediate image of the transformation for further inspection or processing.

Impact on simulation sequence
The DB Staging component impacts the flow of the simulation by first retrieving all data from the original data sources, then acting as a new data source for subsequent components. The component allows overwriting of Read Block Size value of the original source components.

Configuring a DB Staging component

1 Drag the DB Staging component onto the Design window.

2 Connect the DB Staging's 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 the staging table(s).

Specify a valid interface and host name to create the connection. See "DB Staging properties list" on page 141 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 these options:

- Create Staging Table from Input.
- Create Staging Table from Port.

5 If you have selected the Create Staging Table from Input option, select the appropriate Port structure, and click OK. Enter a name for the new table.

If you have selected the Create Staging Table from Port option, enter a name for the new table and select an appropriate port structure. Click Apply.

6 In the Add table window, verify if the information included for the new table is correct and click Create.

7 In the Properties window, click the Stage Options icon to open the Stage Options window.

The Stage Options window also lets you define Truncate Table and the Write Block Size options for each staging table.

8 In the Properties window, click the Query icon and create a query to select data from the staging area.

❖ **Updating port structure**

To update the port structure with changes made to the database:

1 Right-click the DB Staging component.

2 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.

DB Staging properties list

The following tables list the required and optional properties of the Data Staging component. Required properties are displayed in **bold** text.

Required properties

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Specify the data source. Options available on the Host Name list depend on the Interface you select. If you use SQLite as a Host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.
Stage Options	Click the Stage Options icon to set the staging options.

Property	Description
Query	Click the Query icon to create a query that retrieves information from the data source. You can use the Query window to create a simple query, or click the Query Designer icon to generate a query.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.
Read Block Size	Determines the number of records retrieved by the component in a single step.
Pre-processing SQL	Click the Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Specify the database you want to use as data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Select the option to Convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.

Property	Description
Database Options	Click the Database Options icon to set options that override performance defaults and control the behavior of some transactions. See for a list of these properties. See “Database connection settings” on page 94.

DB Staging demos

Sybase IQ ETL includes several demonstrations for the DB Staging component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Staging*, select *DB Staging*.

DemoRepository

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 Data Sink Insert	Adds records to a database table. You can exclude attributes or assign default values to determine the records you insert into the table. Use this component to add all records from the IN-port of the component to a database table.
DB Data Sink Update	Updates or overwrites all records that match a selected key. This component does not insert new records. Use this component to update existing records, not insert new records.

Component	Description
DB Data Sink Delete	DB Data Sink Delete removes records from the destination table that match the incoming values of a selected key. Use this component to delete records from the destination table.
Text Data Sink	Writes transformation results to a text file in a delimited or fixed-length format.
DB Bulk Load Sybase IQ	Uses the Load Table utility to add records to a Sybase IQ table. Use this component to add all records from the IN-port of the component to a table in a Sybase IQ database.

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 146 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. If you want to add a table based on an existing port structure, skip this step. See "Adding a Destination table" on page 145 for additional information.

5 Click Finish.

6 Specify any other optional properties in the Properties window.

❖ **Updating port structure**

To update the port structure with changes made to the database:

1 Right-click the DB Data Sink Insert component.

2 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.

❖ **Loading data at input port**

To load data currently available at the input port to a database or a text file:

1 Right-click the DB Data Sink Insert component.

2 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.

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 DB Data Sink Insert component, and select Add Destination Table from Input.

2 Name the new table. Click OK.

3 In the Add table window, verify if the table information is correct and click Create.

❖ **Adding a Destination Table from an existing port**

1 In the Design window, right-click the DB Data Sink Insert 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 are displayed in **bold** text.

Required properties

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Specify the data source. Options available on the Host Name list depend on the Interface you select. If you use SQLite as a Host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.
Destination Table	Click the Destination Table icon to 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 145.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.

Property	Description
Insert Options	Click the Insert Options icon to determine how attributes will be inserted. The Include columns specify which attributes will be inserted. The SQL expression overwrites the attribute value with the value of the expression at the time of execution. Specify an INSERT statement for each attribute that will overwrite the incoming value of the selected attribute. An example of the SQL INSERT value clause is: `'valid' '[uDate("now")]'` Records will be inserted with all the attributes selected in the Insert Options window. Select or unselect any of the attributes. You can specify values for attributes that will be inserted, thus overwriting the corresponding value of the incoming attribute.
Truncate Table	Select the option to 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	Click the Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to create a query that runs after all components execute. Queries 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	Select the database you want to use as data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.

Property	Description
Standardize Data Format	Select the option to convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Data Sink Insert demos

Sybase IQ ETL includes several demonstrations for the DB Data Sink Insert component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Destination*, select *DB Data Sink - Insert*.

DemoRepository

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.

Use this component to update existing records, not insert new records.

Note If the update values violate any restrictions of the underlying table or object, such as, constraints, referential integrity, 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 151 for specific field requirements.

3 Specify the Destination 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.

You can write to an existing table or add a table based on existing ports in the project. If you want to add a table based on an existing port structure, skip this step. See “Adding a Destination table” on page 150 for additional information.

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**

To update the port structure with changes made to the database:

1 Right-click the DB Data Sink Update component.

2 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.

❖ **Loading data at input port**

To load data currently available at the input port to a database or a text file:

1 Right-click the DB Data Sink Update component.

2 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.

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 DB Data Sink Update component, and select Add Destination Table from Input.

2 Name the new table. Click OK.

❖ **Adding a Destination Table from an existing port**

1 In the Design window, right-click the DB Data Sink Update component, and select Add Destination Table from port.

2 Name the new table.

3 Select the port whose structure you want to assign to the new table.

Note You can also create a Destination table with your own toolset, or select an existing table from the properties list.

DB Data Sink Update properties list

The following tables list the required and optional properties of the DB Data Sink Update component. Required properties are displayed in **bold** text.

Required properties

Property	Description
Interface	Specify the method or driver you want to use to connect to the data source.
Host Name	Specify the data source. Options available on the Host Name list depend on the Interface you select. If you use SQLite as a Host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.
Destination Table	Click the Destination Table icon to 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 150.
Key	Click the Key icon in the Properties window to 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	Specify an authorized database user and a password to protect the database against unauthorized access.

Update Options	Click the Update Options icon to select the attributes (key attributes are not listed) you want to include in the update. By default, all attributes are selected. Unselect those attributes you want 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. The new value can be a constant or an expression.
	In SQL language notation the contents of the columns will be 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. Dynamic expression in square brackets will be evaluated during initialization of the component.

Write Block Size	Specify the number of records to be written to the file or pipe in a single write operation.
Pre-processing SQL	Click the Pre-processing SQL icon to open a window where you can create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to open a window where you can create a query that runs after all components execute. Queries 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 you want to use as data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.

Property	Description
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Select the option to convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Data Sink Update demos

Sybase IQ ETL includes several demonstrations for the DB Data Sink Update component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Destination*, select *DB Data Sink - Update*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer. Repository* | *Projects*, and then select Demo DB Datasink Update.

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 On 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 156 for specific field requirements.

3 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.

You can write to an existing table or add a table based on existing ports in the project. If you want to add a table based on an existing port structure, skip this step. See “Adding a Destination table” on page 155.

4 Click Finish.

5 In the Properties window, click the Key icon to select the columns identifying the records you want 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**

To update the port structure with changes made to the database:

1 Right-click the DB Data Sink Delete component.

2 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.

❖ **Loading data at input port**

To load data currently available at the input port to a database or a text file:

1 Right-click the DB Data Sink Delete component.

2 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.

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 On 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 DB Data Sink Delete component, select Add Destination Table from Input.

2 Name the new table. Click OK.

❖ **Adding a Destination Table from an existing port**

1 In the Design window, right-click the DB Data Sink Delete 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.

Note You can also create a Destination table with your own toolset, or select an existing table from the Properties window.

DB Data Sink Delete properties list

The following tables list the required and optional properties of the DB Data Sink Delete component. Required properties are displayed in **bold** text.

Required properties

Property	Description
Interface	Identifies the method or driver you want to use to connect to the data source.
Host Name	Identifies the data source. Options available on the Host Name list depend on the Interface you select. If you use SQLite as a Host, type the path and file name to the SQLite database file (For example, *c:\mySQLite.db*). If a database with the name you specify does not exist, SQLite will create one.
Destination Table	Click the Destination Table icon to open a window where you can 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 155 for more information.
Key	Click the Key icon in the Properties window to 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, and you can select multiple key columns.

Optional properties

Property	Description
User and Password	Used in combination to identify an authorized database user, and protect the database against unauthorized access.

Property	Description
Write Block Size	Specify the number of records to be written to the file in a single write operation.
Pre-processing SQL	Click the Pre-processing SQL icon to open a window where you can create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to open a window where you can create a query that runs after all components execute. Queries 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	Identifies the database you want to use as data source. If you select this option, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Converts incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to open a window where you can set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

DB Data Sink Delete demos

Sybase IQ ETL includes several demonstrations for the DB Data Sink Delete component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Demonstrations* | *Destination*, select *DB Data Sink - Delete*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer.Repository* | *Projects*, and then select Demo DB Datasink Delete.

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.

2 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 the “Text Data Sink properties list” on page 160 for specific field requirements. Also see:

- Exporting and importing file definitions – 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)– column values for delimited files reflect the current IN-port structure. You can assign a port structure based on another port or recreate the current port structure.
- Working with fixed length files – if you work with fixed-length file types, you must create the columns, and provide the position parameters for each column.

3 Click Save.

4 Specify any other optional properties in the Properties window.

Note If there is an adjacent component available in the Design window, Sybase IQ 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 input port**

To load data currently available at the input port to a database or a text file:

1 Right-click the Text Data Sink component.

2 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. If the buffer is empty, the option will be grayed out.

Exporting and importing file definitions

Export and Import options on the Component window let you save the 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 Double-click Text Data Sink to open the Component window.

2 Click Properties | Export.

3 Select the definition file you want to use.

❖ **Importing file definitions**

1 Double-click Text Data Sink to open the Component window.

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 recreate the current port structure.

❖ **Assigning a new port structure**

1 Double-click Text Data Sink to open the Component window.

2 Click the Assign Port Structure icon in the Column Names panel.

3 Select the port whose structure you want to assign.

❖ **Regenerating column definitions**

1 Double-click Text Data Sink to open the Component window.

2 Right-click the Regenerate the column definition icon in the Column Names panel.

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 Double-click Text Data Sink to open the Component window.

2 Click the Insert a New Attribute icon available in the Column Names panel. You can edit the name of the generated column.

❖ **To remove columns from the output**

1 Double-click Text Data Sink to open the Component window.

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 property labels are displayed in **bold** text.

Required properties

Property	Description
Text Destination	Specify the output file. Text Data Sink prompts you for the destination file when you add the component to the project. To specify a destination file, click the Destination File icon in the from the Properties window, and select an existing file, or type the full path and file name to create one during project execution.
Columns	Click the Columns icon to define columns for data in the source file. If the Property value is not empty, the Columns value reflects the port structure or attribute values you defined on the Component window.

Optional properties

Property	Description
Row Delimiter	Select 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	Select how column 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

Property	Description
Fixed by Bytes	Specify how to interpret the values provided for line length, column start, and column end: • Not selected – the values are interpreted as number of characters. This is the default. • Selected – the values are interpreted as number of bytes.
Encoding	Select an appropriate value from the drop-down menu to set the current character encoding.
Column Header	Select the option to write the column names to the file.
Header	Click the Header icon to create a report header to write to the file. Text Data Sink writes the header before it writes the incoming data. This is an optional output property. Type the header text for a header you want to write. Expressions are allowed in Square Bracket Notation.
Append Data	Select the option to 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 IQ ETL writes to the file in a single write operation.

Text Data Sink Demos

Sybase IQ ETL includes several demonstrations for the Text Data Sink component. These demos are available as Flash Demos on the Help menu and as sample projects in the DemoRepository.

Flash demos

To run the Flash Demo, click *Help* | *Component Demonstrations* | *Destination*, select *Text Data Sink*.

DemoRepository

In the Navigator, select *Repository* | *TRANSFORMER.transformer. Repository* | *Projects*. Then select:

- Demo XML via SQL Data Provider
- Demo Text Data Sink Delimited/Fixed

DB Bulk Load Sybase IQ

DB Bulk Load Sybase IQ is a Destination component that uses the Sybase IQ Load Table utility to add records to a table in an IQ database.

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. See “Adding new Sybase IQ destination tables” on page 164 if you want to write to a new destination table.

5 Click the Load Stage icon. You can:

- 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://<name>`).

  See the “DB Bulk Load Sybase IQ properties list” on page 164 for specific field requirements.

6 Click Finish.

For adding DB Bulk Load IQ to your project, make sure:

- Sybase IQ must be up and running before you add DB Bulk Load IQ to your project. You can increase performance if you run both ETL Server and the IQ database on the same machine. This is not required.
- To connect to the target database on IQ, you need to select a valid host name and interface. See the “DB Bulk Load Sybase IQ properties list” on page 164 for specific field requirements.
- If you want 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 164.

- If you want 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**

To update the port structure with changes made in the database:

1 Right-click the DB Bulk Load IQ component.

2 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.

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 from input**

To create an IQ destination table based on DB Bulk Load Sybase IQ's IN-port:

1 Right-click the DB Bulk Load Sybase IQ component and select Add Destination Table from Input.

2 Enter a name for the new table.

3 Click OK.

❖ **Adding a destination table from an existing port**

To create an IQ destination table based on an available component port:

1 Right-click the DB Bulk Load Sybase IQ 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.

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 you want to use to connect to the data source.
Host Name	Specify the host where the Sybase IQ target is running.
Destination	Click the Destination Table icon to select the destination table from a set of existing tables.
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.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.
Truncate	Select the option to remove all records from the destination table before the load.
Load Script	The Load Table Statement is generated on 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. After you generate the script, you can click Load Script and edit the script.
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, you must enter it here.
Write Block Size	Specify the number of records to be written to the file or pipe in a single write operation.
Pre-processing SQL	Click the Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).

Property	Description
Post-processing SQL	Click the Post-processing SQL icon to create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Specify the database you want to use as data source. Additionally, you must select an appropriate interface, and in some cases, specify an appropriate user ID and password.
Schema	Identifies the schema/owner you want to use as data source. The objects displayed will be restricted accordingly and new tables will be created in that schema.
Standardize Data Format	Select the option to convert incoming `DATE` and `NUMBER` information into a standard format that Sybase IQ 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 with a '.' as the decimal separator.
Database Options	Click the Database Options icon to set options that override performance defaults and control the behavior of some transactions. See “Database connection settings” on page 94.

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 probably 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 set when writing to the data file or pipe, and when generating the LOAD TABLE script. The default values can be specified 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

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.
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

IQ Loader File via Load Table component loads data from a file into a target IQ database via an automatically generated LOAD TABLE statement. This is a self contained component that operates as a data source when it reads from source files and a data sink when it writes to a Sybase IQ database. This component does not have input and output 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 DB via Insert Location properties list” on page 174 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. It includes:

- File Content panel – displays the contents of the source document.
- Properties panel – displays the file description properties.
- Preview panel – displays a tabular view of the data in the source file, based on the currently selected properties.

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 you want to use 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.
Destination	Click the Destination Table icon to select the destination table from a set of existing tables.
Text Source	Identifies 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 168.
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 Alternatively, you can enter a different delimiter.

Optional properties

Property	Description
User and Password	Specify an authorized database user and a password to protect the database against unauthorized access.

Property	Description
Load Script	The Load Table Statement is generated on 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. After you generate the script, you can click Load Script and edit the script.
Truncate	Select the option to remove all records from the destination table before the load.
Pre-processing SQL	Click the Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Post-processing SQL	Click the Post-processing SQL icon to create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Database	Specify the database you want 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	Click the Database Options icon to 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	Select this option when all columns including the last, are delimited by a single ASCII character, to allow the LOAD TABLE command to run in parallel.
Strip	Select the option if you want trailing blanks to be stripped from values before they are inserted. This applies only to variable-length non-binary data.

Property	Description
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 IQ 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.
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 aborts the entire transaction (the default). • 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	Allows the load to continue when it encounters data longer than the limit specified when the word index was created.

IQ Loader DB via Insert Location

IQ Loader DB via Insert Location component loads data from the source database into the target IQ database using the Insert Location statement. This is a self contained component that operates as a data source when it reads from a source database and a data sink when it writes to a Sybase IQ database. This component does not have input and output ports. Therefore, there is no need to create one component for performing the read from the source file 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 Sybase Adaptive Server Enterprise, Oracle, IBM DB2, and Microsoft SQL Server. ETL 4.5.1 supports Sybase ECDA 15.0 with IBM DB2 9.1, Oracle 10g, and Microsoft SQL Server 2005.

  See the Sybase Product Manuals Web site at http://www.sybase.com/support/manuals for Sybase ECDA documentation.

Configuring IQ Loader DB via Insert Location component

1. Drag 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 you want to use as the destination database.
   - Schema – Select schema/owner to restrict the objects displayed and create new tables in that schema.
   - Click Processing to specify Pre-Processing SQL and Post-Processing 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 the Use remote server definition for accessing source database option to get data and metadata from the source. Select this option only if you have defined the source server as a 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.
- 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 specify Pre-Processing SQL and Post-Processing SQL on the source database.
- Select the Create Target Tables option to create the destination tables if they do not exist.
- Select the Continue on Error option if you want the 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.

- 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 table(s) 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 table(s) from the drop-down menu. There should be a one on one mapping (one source for each destination) between sources and destinations.

6 Click the Truncate Destination option 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

Whenever 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 LC_ALL is not set, 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” of the *Sybase IQ 12.7 System Administration Guide*.

IQ Loader DB via Insert Location properties list

IQ Loader DB via Insert Location properties list identifies the connection parameters and other items you need to 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.
Source Host Name	Specify the data source.
Source Transfer List	Specify the schema qualified source table name and the target table name. In the target truncate column, specify 1 if you want the target table to be truncated, or else 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	Click the IQ Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
IQ Post-processing SQL	Click the IQ Post-processing SQL icon to create a query that runs after all components execute. Queries 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 Database	Specify the source database.
Source Schema	Specify the schema/owner to restrict the objects displayed.
Source Pre-processing SQL	Click the Source Pre-processing SQL icon to create a query that runs during component initialization. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).

Property	Description
Source Post-processing SQL	Click the Source Post-processing SQL icon to create a query that runs after all components execute. Queries can include one or more SQL statements. If you use multiple statements, separate them with a semicolon (;).
Create Target Tables	Select the option to create the destination tables if they do not exist.
Continue on Error	Select the option to continue execution even if an error occurs when loading data into the database.
Limit Rows	Specify the maximum number of rows you want 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 table(s) for a load. The default is 0.
Encrypted Password	Select the option if you want the password to be transmitted in encrypted format.
Network Packet Size	Specify the network packet size
Load Script	The Insert Location statements are generated on 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. After you generate the script, you can click Load Script and edit the script.

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	Controls job flow execution. 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 non-critical. Critical project failures cause the Synchronizer to signal failure.
Multi-Project	Provides a visual representation of the project groups within the job. Multi-Project combines the properties of the Project and Synchronizer components. Use this component when your job consists of a large amount of independent projects, in other words, projects that can be executed in any order.
Finish	Represents the end of a successful job execution. Use this component to mark the successful end of a job.
Error	Provides a visual representation of the end of a failed job execution. 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 the Project component to the job

❖ **Adding and configuring the Project component**

1 Drag the Project component from the Component store onto the Design window to add it to a job.

2 Connect it with its adjacent components.

3 Double-click the component and select a project to execute.

Required properties

Property	Description
Project Name	Click the Project Name icon to select the project you want to add.

Optional properties

Property	Description
Continue on DB Write Errors	Select the option if you want the project execution to continue even if an error occurs on loading data into a database via a DB Data Sink component. If errors have been ignored due to this option the project will be stated as 'failed'. Combined with the Reject Log (see “Database connection settings” on page 94) 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 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 non-critical. Critical project failures causes the job flow to follow the branch at the Error port of the Synchronizer.

The Success and Error ports of the Synchronizer component can either be connected to:

- Multiple Project components.
- Multiple Multi-Project components.
- Multiple Project and Multi-Project components.
- Single Finish component or Error component.

❖ **Adding and configuring the Synchronizer component**

1 Add the component to the job and connect it with all the projects that signal their execution status to this component.

2 (Optional) In the Property window click the Synchronize Options icon to select critical projects.

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

Multi-Project provides a visual representation of the project groups within the job. Multi-Project combines the properties of the Project and Synchronizer components. The Success and Error ports of the Multi-Project component can either be connected to:

- 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, in other words, projects that can be executed in any order, even in parallel when used in multi-engine jobs.

❖ **Configuring the Multi-Project component**

1 Add the component to the job and connect it with its adjacent component.

2 In the Properties window, click the Projects Execution icon to select the projects to execute.

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. Trying to add a project multiple times will have no effect.

The options available for projects 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 will continue even if an error occurs when loading data into a database.
- **Critical** – You can define each single project as critical or non-critical. 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. Finish can be connected to the following 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. Error can be connected to the following 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

This chapter provides information about how to use the Sybase ETL Server. See “Sybase IQ ETL architecture” on page 2 for more information.

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 IQ ETL Development. Sybase ETL Server uses UDP broadcasts to inform other servers about urgent events, such as start and stop, as well as system failure or crash.

The default port for communication is 5124. You can change the default port in the *INI* file or using the command line.

All communication between the 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 the Sybase ETL Server

At a command prompt, enter:

On Windows:

```
GridNode
GridNode --port 5500
```

On Linux and UNIX:

```
GridNode.sh
GridNode.sh --port 5500
```

Stopping the 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]
```

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.

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: Command line parameters

Command	Abbreviation	UNIX	Windows	Description
install	inst	yes	yes	Installs the application as a Unix daemon or Windows service.
remove	rm	yes	yes	Removes daemon or system service start.

Command	Abbreviation	UNIX	Windows	Description
setoptions	so	no	yes	Sets the command line options to be used when running as a Windows service.
getoptions	go	no	yes	Prints the command line options to be used when running as a Windows service.
background	bg	yes	no	Sets background processes that do not overuse system resources.
no_pidfile	nopid	yes	no	Sets the file to which the server records the process ID of the daemon.
console	con	yes	yes	Writes detailed error information and trace messages on the console.
diagnosis	diag	yes	yes	Lists application environment.
tracelevel	tl	yes	yes	Sets trace level for debugging from 0 (no trace) to 5 (very verbose).
server	s	yes	yes	Identifies the remote server to be used.
port	p	yes	yes	Identifies the port number to operate on.
version	V	yes	yes	Identifies the application version information.
help	h	yes	yes	Displays help information.
licenses	ll	yes	yes	Identifies the short information about available licenses and their status.
nodelist	nl	yes	yes	Lists all known peer nodes.
shutdown	sh	yes	yes	Shuts down the node.
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.

Command	Abbreviation	UNIX	Windows	Description
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.
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.

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 185. 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", database interface is "dbodbc", host name is "etl_comp", client is "transformer", user is "TRANSFORMER", and the encrypted version of the password is "1234ABCD".

Additionally, you can also use these command line parameters for executing 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 use 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 line, type in the commands, parameters, and values in a continuous line. The examples in this section are split over lines for clarity.

In Sybase IQ ETL versions earlier than 4.5, the execution of projects and job was done using the ProcessQ application, which is distributed with ETL Server and used internally on Windows platform only. ProcessQ is now deprecated and provided for backward compatibility only. Many of the ProcessQ parameters have been disabled and can no longer be used.

Note Sybase recommends you not to use ProcessQ to execute projects and jobs.

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 (HTTP, HTTPS, FTP, FTPS) by using the keys “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 5 provides the most detailed information. During normal execution, set this value to 0 to minimize the affect on performance.
Logging	Flushtime	1-n	1	Indicates seconds between the internal log flashes.

Troubleshooting Sybase ETL Server

Before you contact Technical Support:

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.

Syntax:

GridNode --version

Output:

```
Sybase IQ ETL Server(4.5.0.24797)

Copyright – 2003 - 2008 Sybase, Inc.  All rights reserved.
Unpublished rights reserved under U.S. copyright laws. This
product is the confidential and trade secret information
of Sybase, Inc.  Use, duplication or disclosure of the
software and documentation by the U.S.Government is
subject to restrictions set forth in a license agreement
between the Government and Sybase, Inc. or other written
agreement specifying the Government's rights to use the
software and any applicable FAR provisions, for example,
FAR 52.227-19.Copyright (c) Sybase, Inc. 2002-2006

Grid Node:
Central Library V 1.0.0.0
```

5 Verify the available licenses.

Syntax:

GridNode --licenses

Output:

```
Sybase IQ ETL Server (4.5.0.24797)

Copyright – 2003 - 2008 Sybase, Inc.  All rights reserved.
Unpublished rights reserved under U.S. copyright laws. This
product is the confidential and trade secret information
of Sybase, Inc.  Use, duplication or disclosure of the
software and documentation by the U.S. Government is
subject to restrictions set forth in a license agreement
between the Government and Sybase, Inc. or other written
agreement specifying the Government's rights to use the
software and any applicable FAR provisions, for example,
FAR 52.227-19.Copyright (c) Sybase, Inc. 2002-2006

Grid Node
-------------------------------------------------
Product ID: SybaseIQETLServer
Machine ID: 9TuA+igB6298Hys=
-------------------------------------------------
File: sybase_iq_etl_server.license
Product: Sybase IQ ETL Server (SybaseIQETLServer)
Version: 4.5
License: IQ ETL Platforms 4.5
```

```
Status: Valid
```

APPENDIX A

Function Reference

This appendix provides a reference for the Sybase IQ ETL functions.

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
uBitXOr	Bitwise Exclusive OR operation
uBitNot	Bitwise Not 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"
```

uBitXOr

Description: Bitwise Exclusive OR operation

Syntax: number uBitXOr(value1 , value2)

Parameters: *number value1, value2*
Values to calculate

Examples:
```
uBitXOr(10, 3) // returns "9"
```

uBitNot

Description: Bitwise Not operation

Syntax: number uBitNot(value)

Parameters: *number value*
A numeric argument

Examples:
```
uBitNot(10) // returns "-11"
```

Boolean

Function	Description
uIsAscending	Returns 1 if every parameter is equal or greater than its predecessor
uIsBoolean	Returns 1 if the parameter is one of 1, true, or yes
uIsDate	Returns 1 if the parameter can be interpreted as a date
uIsDescending	Returns 1 if every parameter is equal 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
uNot	Returns 0 for an input of 1 and 1 for an input of 0

uIsAscending

Description: Returns 1 if every parameter is equal or greater than its predecessor

Syntax: number uIsAscending(params, ...)

Parameters:

params

A list of expressions or values of any data type.

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:

- one of 1, true, or yes
- 0, no, or false

Returns 1 if the parameter is

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 the following 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, refer to 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 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 an 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
```

uNot

Description: Returns 0 for an input of 1 and 1 for an input of 0. This function is only used in conjunction with the uIs-Functions, because the Boolean values returned are not true and false, but are 0 and 1.

Syntax: number uNot(expression)

Parameters: *expression*
A numeric value which should be negated

Examples:

```
uNot(1) // 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 Hexadecimal value into Integer value
uToHex	Converts Integer value into hexadecimal digits
uHexDecode	Composes a string from hexadecimal values
uHexEncode	Encodes the characters of a string into hexadecimal notation

Function	Description
uToUnicode	Converts a string into its Unicode representation
uURIDecode	Decodes a string replacing the escape sequences with their original values
uURIEncode	Replaces certain characters in an 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

```
uBase64Dcode("QQAgAHMAZQA=") // returns "A secret="
```

uBase64Encode

Description — Encodes a string into a Base64 representation

Syntax — string uBase64Encode(input)

Parameters — *string input*
The string to encode.

Examples

```
uBase64DEncode("A secret") // returns "QQAgAHMAZQA="
```

uConvertDate

Description — Converts a date string into the default or a custom date format. 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 string (see the following table). The outputformat parameter is optional, and if it is omitted, the date is converted in 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 will be 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"
```

Usage

Description

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 will be converted using the format `y-M-D H:N:S`. The function handles dates from *1582* to present day. If the date could not be converted, the result string will be empty.

Identifier Description

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
`S`	seconds
`s`	hundredth of seconds
`t`	thousandth of seconds

Identifier	Description
A	AM/PM
d	day of week (5)
D	day of week (Friday)
E	day of year (001..366)
G	week of year (01..52)
F	week of month (1..6)

uFromHex

Description: Converts Hexadecimal value into 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 Integer value into hexadecimal digits

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 hex 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 hex 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 the 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 an 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 only return 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 206
- "Modifiers" on page 207
- "Date and time calculations" on page 208
- "Known limitations" on page 209
- "Date and time function list" on page 209

Time Strings

Description — A time string can be in any of the following 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 through 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 as follows:

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

The first size modifiers (1 through 6) simply add the specified amount of time to the date specified by the preceding time string.

The "start of" modifiers (7 through 9) shift the date backwards 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 converted to julian day numbers easily.

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 timezone

```
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. In this implementation, 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 206 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 which contains the first Thursday of the calendar year, which implies that it is also:

- The first week which 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 which contains the last Thursday of the Calendar year
- The last week which 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-
0102:34:56')*86400
```

Compute the number of days since the battle of Hastings

```
uJuliandate('now') - uJuliandate('1066-10-
14','gregorian')
```

Compute the date and time given a UNIX timestamp 1092941466, and compensate for your local timezone

```
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 — To get the name of month from a date

```
uMonthName("now") // returns current name of month
```

Setting the locale to English

```
uSetLocale("English")
uMonthName("1969-03-13 10:22:23.231") // returns
"March"
```

Setting 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.
```

Setting the locale to English

```
uSetLocale("English")
uMonthNameShort("1969-03-13 10:22:23.231") // returns
"Mar"
```

Setting 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
```

Setting the locale to English

```
uSetLocale("English")
uWeekdayName("1969-03-13 10:22:23.231") // returns
"Thursday"
```

Setting 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
```

Setting the locale to English

```
uSetLocale("English")
uWeekdayNameShort("1969-03-13 10:22:23.231") //returns
"Thu"
```

Setting 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"
```

Errorhandling

Function	Description
uError	Writes an error text into log and signal an error
uErrortext	Returns the last error message
uWarning	Writes a warning message into log
uInfo	Writes an informal message into log
uTrace	Writes a trace message into log
uTracelevel	Sets the detail level of trace messages in the log

uError

Description — Writes an error text into 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 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 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 log

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 Heavy logging slows execution dramatically.

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 whole path to the file if it exists or an empty string if it does not exist. Is infotype set to `SIZE`, the size of the file will be returned or an empty string if the file does not exist.

Note Be aware that you have to double the 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

Getting 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*, means the whole 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

Accessing 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 IQ 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 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. See *Usage* below for a list of special escape sequences in the format string will be replaced by the referring date part.

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

The function uFormatDate returns a user defined string with date information. Special escape sequences in the format string will be replaced by the referring date part.

Escape sequences

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
%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 values case sensitive 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 values case sensitive 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 the [...] and [^...] matching, a ']' character can be included in the list by making it the first character after '[' or '^'. A range of characters can be specified using '-'.

Example:

"[a-z]" matches any single lower-case letter. To match a '-', make it the last character in the list.

To match '*' or '?', put them in "[]".

Example: abc[*]xyz, matches "abc*xyz" only.

uLike

Description

Compares values case insensitive.

The uLike function does 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 it's lower/upper case equivalent (For example, case-insensitive matching).

Note Currently uLike only understands upper/lower case for 7-bit Latin characters, hence the uLike is case sensitive for 8-bit ISO8859 characters or UTF-8 characters. For example:
`uLike('a' ,'A')` is 1
`uLike('æ' ,'Æ')` is 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]*")  // return 0
uMatches("1.23","[-+]?[0-9]*\\.?[0-9]*")  // return 1
```

Lookup

Function	Description
uChoice	Returns the value of a given parameter specified by an index
uFirstDifferent	Returns the first parameter value which is different 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
Index number referencing the return value. Zero based.

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 correcponding color name

```
uChoice(IN.Color, "n.a.", "Red", "Blue", "Green")
```

uFirstDifferent

Description: Returns the first parameter value which is different 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) will be 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 taken as the delimiter.

Syntax — string uToken(text, index [, delimiter])

Parameters

string text
A string to investigate

Integer index
Number of token 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."
```

Miscelleanous

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 amount 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. The following formats are available:

- *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 list below)

string country (optional)
Country name to be used (see list below)

string codepage (optional)
Codepage 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 following language strings are recognized. Any language not supported by the operating system is not accepted by uSetLocale.

Note The three-letter language-string codes are only valid 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"

Primary Language	Sublanguage	Language String
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"
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"

Primary Language	Sublanguage	Language String
Turkish	Turkish	"trk" or "turkish"

Country/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/region-name codes are from ISO/IEC (International Organization for Standardization, International Electrotechnical Commission) specification 3166.

Country/Region	Country/Region String
Australia	"aus" or "australia"
Austria	"austria" or "aut"
Belgium	"bel" or "belgium"
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"

Country/Region	Country/Region String
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 amount of milliseconds

Syntax — string uSleep(msecs)

Parameters — *integer msecs*
Number of milliseconds to suspend

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 — The uSystemFolder function can be used to access a special directory on the filesystem. You can specify the following folders:

Group	Name	Description
Application	APP_MAIN	The base application path. A typical path is *C:\Program Files\ETL*

Group	Name	Description
Application	APP_LIB	Shared library directory, typically in the lib folder of the applications directory
Application	APP_LOG	Shared library directory, typically in the lib folder of the applications directory
Application	APP_CONFIG	Config file directory, typically in the etc folder of the applications directory
Application	APP_LICENSE	License directory, typically in the license folder of the applications directory
Application	APP_SCRIPT	Script directory, typically in the scripts folder of the applications directory
Application	APP_GRAMMAR	Grammar directory, typically in the grammar folder of the applications directory
Application	APP_LANGUAGE	Language file directory, typically in the language folder of the applications directory
Application	APP_DATABASE	Database directory, typically in the database folder of the applications directory
Application	APP_TEMP	Temporary directory, typically in the temp folder of the applications directory
Application	APP_DEMODATA	Demodata directory, typically in the demodata folder of the applications directory
Application	APP_USERDATA	Directory where userspcecific file will be stored. Typical path is *C:\Documents and Settings\username\Application Data\ETL*
Windows	ALTSTARTUP	The file system directory that corresponds to the user's nonlocalized 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*.

Group	Name	Description
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*.
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.

Group	Name	Description
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*.
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*.

Group	Name	Description
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*.
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 put their 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*.

Group	Name	Description
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 onto 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*.
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 mail to a SMTP server

uHostname

Description Returns the local network name

Syntax string uHostname()

Examples

```
uHostname()  // returns something like "pollux"
```

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, username and password to use

string sender
Email address of sender.

string recipients
Comma separated list of recipients

string subject
Subject of the mail message

string body
The content of the email message

Examples

```
uSMTP("Just a mail")

uSMTP("Testmail!", "Just a mail")
```

Usage

The uSMTP function allows sending emails to multiple recipients using a SMTP server.

Specifying the SMTP server, port, user and password

The server URL for specifying the SMTP server has the following 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 and Password – Username and Password will be used in order to authenticate the client. If not provided, no authentication will be made.

Note If the username contains a @ sign, you have to replace it with # to avoid ambiguities.

Port – The TCP port to be used, default is 25.

URL Examples

```
myServer
myServer:123
SMTPS://myServer:123
Me:secret@myServer
```

Specifying recipients – You can add a list of addresses separated by a comma. By default, all recipients are addressed directly. In order to specify a CC or BC just prefix the 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>
```

Security – If the SMTP server allows communicating using an encrypted connection, this will be done automatically. If username and password is provided, authentication methods are tried in the following sequence: PLAIN, LOGIN.

Custom Defaults – 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>
```

INI File 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

Function	Description
uCeil	Returns least integer greater than or equal to argument
uDiv	Returns the division integer
uExp	Returns the exponential, base e
uFloor	Returns greatest integer less than or equal to argument
uLn	Returns the natural logarithm (base e) of a number
uLog	Returns the logarithm of a number
uMod	Returns the modulo of division
uPow, uPower	Returns the value of a base expression taken to a specified power
uRandom	Returns a random number
uRound	Returns the rounded argument to nearest integer
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
Rounding 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 will be used

Examples

```
uLog(100) // returns 3
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 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: Evaluation functional expressions:

```
uEvaluate("3 + 5”)

uEvaluate("parseFloat(IN.Salary) + 1500")
```

Defining custom functions:

```
uEvaluate("function timesTwo(a){ return a*2; }")
```

Using custom functions:

```
uEvaluate("timesTwo(4)")

uEvaluate("timesTwo(IN.Salary)”)
```

Evaluating scripts:

```
uEvaluate("if ("parseFloat(IN.Salary) > 2000") {2000;}
else {("parseFloat(IN.Salary) + 500");}")
```

String

Function	Description
uAsc, uUnicode	Returns unicode character value of a specified character

Function	Description
uChr, uUniChr	Returns the Unicode string responding the 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	Find the first position of a substring within a string
uLower, uLow	Returns the input string in lower case letters
uLStuff	Fills the left side of a string up to 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	Find the last position of a substring within a string
uRStuff	Fills the right side of a string up to 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 upper case 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 responding the 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 and every 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) indicating if we allow empty fields

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 amount 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 space (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 that will 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 chars 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"
```

Operators

Function	Description
uEQ	Returns 1 if both parameters are equal and no parameter is NULL
uNE	Returns 1 if both parameters are not equal and no parameter is NULL
uGT	Returns 1 if the first parameter is greater than the second parameter and no parameter is NULL
uGE	Returns 1 if the first parameter is greater or equal than the second parameter and no parameter is NULL
uLT	Returns 1 if the first parameter is lower than the second parameter and no parameter is NULL
uLE	Returns 1 if the first parameter is lower than or equal the second parameter and no parameter is NULL

Note For compatibility reasons, operators are also provided as functions.

uEQ

Description: Returns 1 if both parameters are equal and no parameter is NULL

Syntax: number uEQ(value1, value2)

Parameters: *value1, value2*
Numeric or string values to compare

Examples:
```
uEQ(1,2)    // returns 0
uEQ(1,1)    // returns 1
uEQ(null,1) // returns 0
```

uNE

Description: Returns 1 if both parameters are not equal and no parameter is NULL

Syntax: number uNE(value1, value2)

Parameters: *value1*, *value2*
Numeric or string values to compare

Examples:

```
uNE(1,2)    // returns 1
uNE(1,1)    // returns 0
uNE(null,1) // returns 0
```

uGT

Description: Returns 1 if the first parameter is greater than the second parameter and no parameter is NULL

Syntax: number uGT(value1, value2)

Parameters: *value1*, *value2*
Numeric or string values to compare

Examples:

```
uGT(2,1)    // returns 1
uGT(1,2)    // returns 0
uGT(1,1)    // returns 0
uGT(null,1) // returns 0
```

uGE

Description: Returns 1 if the first parameter is greater or equal than the second parameter and no parameter is NULL

Syntax: number uGE(value1, value2)

Parameters: *value1*, *value2*
Numeric or string values to compare

Examples:

```
uGE(2,1)    // returns 1
uGE(1,2)    // returns 0
```

```
uGE(1,1)    // returns 1
uGE(null,1) // returns 0
```

uLT

Description
Returns 1 if the first parameter is lower than the second parameter and no parameter is NULL

Syntax
number uLT(value1, value2)

Parameters
value1, *value2*
Numeric or string values to compare

Examples

```
uLT(2,1)    // returns 0
uLT(1,2)    // returns 1
uLT(1,1)    // returns 0
uLT(null,1) // returns 0
```

uLE

Description
Returns 1 if the first parameter is lower than or equal the second parameter and no parameter is NULL

Syntax
number uLE(value1, value2)

Parameters
value1, *value2*
Numeric or string values to compare

Examples

```
uLE(2,1)    // returns 0
uLE(1,2)    // returns 1
uLE(1,1)    // returns 1
uLE(null,1) // returns 0
```

Trigonometric

Function	Description
uAcos	Returns the arccosine (in radians) of a number
uAsin	Returns the arcsine (in radians) of a number
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. It also provides additional information for some of the supported interfaces.

Interface specific database options

DB Option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Allow fallback	1 (Not used)	1 (Not used)	1	1	1 (Not used)	1 (Not used)	If set to 1, standard loading operations are used when bulk operations are not available.
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 compatability.
Auto commit	Not available	1 If database is IQ or ODBC driver is ASA ODBC 9, ODBC uses 0 instead of the user input.	0	0	Not available	Not available	Puts the connection into auto commit mode.

DB Option	Interface and default value						Description
	Oracle	**ODBC/DB2**	**Sybase 15**	**Sybase**	**OLEDB**	**SQLite Persistent**	
Auto vacuum	Not available	Not available	Not available	Not available	Not available	0	Reclaims the space when objects are deleted from the database.
Block inserts	Not available	auto Bulk operations are used only if the ODBC driver is ASA9, value is auto, there is no LOB type, and the option "Write rejected records to file" is null.	Not available	Not available	Not available	Not available	Use block wise binding when preparing a SQL statement for execution.
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 LOB's 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	Character set to use with CTLIB (user defined).
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.

DB Option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Connect timeout	0 (Not used)	0 (Not used)	0	0	10	0	Stops trying to connect after the 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 uses 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.
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 bulk load	1 (Not used)	1/Not used	1	1	1 (Not used)	1	If set to 1, bulk operations are used, when applicable.
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, columns names will be fully qualified, following this pattern: <table-name/alias> <column-name>.

DB Option	Interface and default value						Description
	Oracle	**ODBC/DB2**	**Sybase 15**	**Sybase**	**OLEDB**	**SQLite Persistent**	
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. This is used to ensure 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.
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.
LOB truncate size	Not available	1024 (Not used)	10000	10000	Not available	Not available	LOB is truncated when it exceeds LOB truncate size.
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.

DB Option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
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.
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, else they are refered 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.

DB Option	Interface and default value						Description
	Oracle	**ODBC/DB2**	**Sybase 15**	**Sybase**	**OLEDB**	**SQLite Persistent**	
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, will pause to write at critical moments but not as frequently as full. If you select Off = 0, data is handed off to operating system and immediately 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 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.
Truncate reject log	1	1	1	1	1	1	Log truncates on database connect. A value of 0 appends data to an existing log file.
Use DELETE instead of TRUNCATE	0	0	0	0	0	0	If set to 1, DELETE statement is used instead of TRUNCATE.
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.

DB Option	Interface and default value						Description
	Oracle	ODBC/DB2	Sybase 15	Sybase	OLEDB	SQLite Persistent	
Write error code to reject log	1	1	1	1	1	1	Adds error code to the reject log.
Write error text to reject log	1	1	1	1	1	1	Adds error text to the reject log.
Write header to reject log	0	0	0	0	0	0	Specifies whether the reject log should contain a header line.
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.

Note:

- 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.

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
Sybase	ASE ASA IQ	IQ	IQ	ASE ASA IQ	IQ	IQ
ODBC	Any data source that can be accessed via ODBC.	IQ MS-Access	IQ	MS-Access IQ ASA ASE	IQ	unsupported
OLE DB	IQ SQL-Server	IQ	IQ	IQ	unsupported	unsupported
Oracle	Any Oracle database system that can be accessed by Oracle Call Interface (OCI).	unsupported	unsupported	unsupported	unsupported	unsupported

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
IBM DB/2	Any DB2 database system that can be accessed by the IBM DB/2 client interface.	unsupported	unsupported	unsupported	unsupported	unsupported
SQLite Persistent	Any SQLite database file.	unsupported	unsupported	unsupported	unsupported	unsupported

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 to install one of the supported versions for your interfacing driver. See “Interface support” in the *Sybase IQ ETL 4.5.1 Release Bulletin* for a list of all interface driver versions that Sybase IQ ETL 4.5.1 supports.

Working with the SQLite Persistent interface

Sybase IQ ETL technology includes a built-in, general purpose, relational database to be used for temporary data storage and staging. It is based on SQLite, a very fast, widely used, mostly SQL-92 compliant database. SQLite is a small C library that implements a self-contained, embeddable, zero configuration SQL database engine.

Note Use SQLite for your test environment only. Do not use it in a production environment.

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 an 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. If you want 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.

Example – to create a new SQLite database file *mySQLite.db* or to connect to an existing *mySQLite.db* database file:

- Interface: SQLite Persistent
- Host Name: mySQLite

The database file created is named *mySQLite.db*.

Creating a SQLite table

You can 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. Also, you can 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 IQ ETL datatypes to Oracle datatypes.

Table B-2: Data type mapping from Oracle interface to Sybase IQ 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 SCD

In Type 1 SCD, new data overwrites existing data. The existing data is lost and there is no tracking of historical changes. Type 1 SCD is easy to maintain but is useful only if you need not track the historical changes.

Example: 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 SCD

Type 2 SCD 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 expiration date to identify the time period for which the record was active. Use type 2 SCD to keep a full history of dimension data in the table.

Example: See “Case study scenario” on page 274.

Type 3 SCD

Type 3 SCD 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 SCD when you need to track historical changes that occur only for a finite number of time.

Example: You have 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 SCD, 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 IQ 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, 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

PRODUCT table is updated again on 28 July 2008:

A product is removed from the table – Mouse.

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 SCD 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, but is not part of the use case implementation.

- 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 274.

  Before executing this project, execute the SCD - Initial Load project. To accomplish a full update, also execute the SCD - Update Deleted project.

  Note When the ‘Demo Product Price SCD - Initial Load’ project is executed for the first time, all source records are processed as new records in target tables that are empty.

- 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 274.

  Before executing this project, execute the SCD - Initial Load project. 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 individually 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. The methods described 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. For a list of Lookup components, see “Lookup components” on page 131.

In case you need to perform calculations on the data to be transferred, consider to use the lookup functionality of the Data Calculator component. See “Data Calculator JavaScript” on page 115.

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 an appropriate return value has to be selected as well.

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 91. 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. Exists will be 1 (lookup value) for all records existing in target and 0 (default value) for all non-existing 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. 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 table is checked for each source record based on the source key attribute, and the values are compared.

a Choose an appropriate Lookup component.

As this requires to look up multiple values for a single key attribute and perform comparisons, use the Data Calculator component. See “Data Calculator JavaScript” on page 115.

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. For a list of Data Provider components, 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 requires to 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 91.

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. For a list of Data Provider components, 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. For a list of Lookup components, see “Lookup components” on page 131.

Use the lookup functionality of the Data Calculator component if your source data does not reside in a database. See “Data Calculator JavaScript” on page 115.

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 an appropriate return value must be selected 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 or still exists. 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 91.

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. Removed will be 0 (lookup value) for all existing source records and 1 (default value) for all non-existing records.

Alternatives

- If the source is a database that provides ascending indicator for insertions, updates, or deletions (like auto increments, 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 139.

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. For more information on the Data Splitter component, see “Data Splitter JavaScript” on page 124.

To remove records from the data stream, define conditions for every output 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 output port by deleting one of the default output 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 can be constants or the result of an expression. See “DB Data Sink Insert” on page 144 and “DB Data Sink Update” on page 149.

In the ETL demo projects and jobs, the insert options use the:

- Expression – '[uDate('now','localtime')]' (today) for the Valid From date attribute.
- Constant – '9999-12-31' for the Valid To date attribute.

The update options use the expression:
'[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 updating a subset of attributes instead of updating the complete record. To exclude attributes from update, you unselect them in the update options window. See “DB Data Sink Update” on page 149.

In the ETL demo projects, old records are outdated by updating the Valid To date attribute only.

Index

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X