SYBASE®

Users Guide

Sybase ETL Small Business Edition

4.2

[Windows]

DOCUMENT ID: DC00736-01-0420-01

LAST REVISED: April 2007

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

Contents

About This Book

Audience

This guide is for users of Sybase ETL Small Business Edition, which provides extract, transform, and load (ETL) capabilities that you can use to transform data from data providers to data targets. Sybase ETL SBE Development supports a variety of transformation capabilities and enables you to convert, cleanse, merge, and split data streams. The resulting data stream can then insert, update, or delete data in a given data target. You view the shape of your data at any step of the data transformation process by using the unique Sybase ETL "Step and See" technology.

How to use this book

This book contains the following chapters:

- Chapter 1, "Sybase ETL SBE" gives you a brief overview of Sybase ETL SBE architecture and the feature set of Sybase ETL Development and Sybase ETL Server.
- Chapter 2, "Sybase ETL SBE Desktop" helps you become familiar with the design environment for Sybase ETL SBE projects and jobs. This chapter explains the Sybase ETL SBE desktop, toolbars, and general GUI functionality.
- Chapter 3, "Getting Started" helps you get started with Sybase ETL SBE.
- Chapter 4, "Projects and Jobs" guides you through the process of creating, simulating, and executing projects and jobs. It provides valuable insights related to simulation mode. It also shows you how to use templates to create projects and jobs for special purposes.
- Chapter 5, "Advanced Concepts and Tools" describes the set of built-in tools that make your design work easier and faster. Familiarize yourself with Sybase ETL SBE tools, such as Content Explorer or the Query Designer, to speed up the design of your project work.
- Chapter 6, "Components" describes Sybase ETL SBE component concepts and all the information relevant to every component.
- Appendix A, "Function Reference" describes the built-in set of functions available in Sybase ETL SBE.
- Appendix B, "Sybase ETL Server" describes usage and architecture of the Sybase ETL SBE application.

- Appendix C, "Queuing and Executing Process Calls" describes ProcessQ, which is an application that can queue and execute process calls in parallel or in sequence.

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

If you need help

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

CHAPTER 1

Sybase ETL SBE

Sybase ETL SBE architecture

When you install Sybase ETL SBE on Windows, you install Sybase ETL SBE Development and Sybase ETL Server.

Sybase ETL SBE Development

Sybase ETL SBE Development is divided into two major components:

- Sybase ETL SBE Development desktop – provides the graphical user interface. Use the desktop to create and design data transformation projects.
- Sybase ETL SBE Development engine – controls the actual processing, such as connecting to databases and executing procedures.

Sybase ETL Server

The Sybase ETL Server provides the GRID engine service, which, by default, processes desktop requests locally.The GRID engine leverages JavaScript to extend the transformation framework across multiple operating systems.

For more information about Sybase ETL Server and GRID engines, see Appendix B, “Sybase ETL Server.”

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 the correct data sources where they read and transform data. A project consists of various components that can be freely arranged on your project desktop. You can add components to your project simply by dragging them from a section of the Component Store onto your workspace.

Multiple projects can be run sequentially or parallel in a **job**. Jobs control the order in which projects are being executed. Jobs can be scheduled and monitored.

Component concepts

Stepping a component record-by-record

In simulation mode, many of the transformation components offer a convenient way to step through the current set of data and visualize the result of any applied transformation immediately.

Component variables and ports

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 mapping of port structures by applying a mapping on the link that connects two components.

Adaptable port structure and mapping

When adding and connecting components to a project, Sybase ETL SBE normally tries to create a standard mapping between an OUT-port and an IN-port, depending on the object definition of data providers and data sinks. You can modify the mapping by adding attributes to the port structure that can be referenced immediately inside the component.

Running projects and jobs

Use simulation and execution modes to run a project.

Simulation and execution perform all functions of the components included in the simulated project, including the physical transfer of data into the respective data targets (data sinks).

During simulation you can step through the project component by component. The data flow is visible on any link and within any component included. From simulation mode, you can inspect any component and modify mappings and calculations. After making changes, you can re-initialize the component with the new settings and step to the next component. There is no need to start the simulation from the very beginning of the project after any of the components have been changed.

Jobs can be executed from the desktop or as a scheduled task.

Customizing a project

You can create data transformation projects without manually entering a single line of programming code or SQL statement, for example:

- To generate SELECT statements inside Queries, Lookup Definitions, Pre- and Post Processing SQL, use the Query Designer.
- To freely map attributes between data sources and data sinks, use the data mapping features of the links between the components.
- To create temporary or persistent staging tables, use the built-in Create Table from port command of the respective component.
- To create tables in the destination database, use the built-in Create Table from port command of the respective component.
- To browse both schema information and data content of all connected data sources, use the Content Explorer.
- To read hierarchical XML documents and generate a relational structure automatically, use the XML via SQL component.
- To schedule the execution of your projects, you can create Jobs within the Sybase ETL Small Business Edition desktop.

To deal with complex data transformation requirements, familiarize yourself with Sybase ETL Small Business Edition concepts.

For example, you can:

- Use manually optimized SQL SELECT statements to adjust and fine-tune your data extraction process.
- Use SQL to apply data manipulation commands inside pre- and post-processing commands.

- Use JavaScript to write procedures, do complex calculations, or manipulate objects in the operating system environment.
- Use indirection in expressions (Square Bracket Notation) to assign values dynamically to expressions to control your projects by using environment or user variables.

Understanding repositories

The repository contains all data and information related to Sybase ETL SBE objects, projects, and jobs.

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.

Converting datatypes and data formats

Datatypes as originated in the data source are preserved during the transformation process.

Internally, Sybase ETL SBE distinguishes string and numeric datatypes. The Standardize Data Format option of the Data Providers or Data Sinks automatically converts the data to a standard format that then is automatically converted to a format the target database understands. Therefore, you do not have to deal with the various date and number formats when working with different databases.

This setting is activated by default. However, if you are experiencing problems with date or number fields, you can disable this setting on the component that causes trouble and convert the data manually.

SQL

Most of the data delivered by data providers is defined by using SQL stored in the Query property. Sybase ETL Small Business Edition supports a modified set of the SQL-92 standard.

If you do not want to get into the details of SQL-92, you can use the built-in Query Designer to draw the query and have the SQL generated for you. If you want to write SQL manually, or copy SQL from existing projects, you can manually specify your SQL for the Query property.

Tools

Structural and catalog information from all connected data sources is accessible through Sybase ETL Small Business Edition tools, such as the Content Explorer and the Query Designer. You can browse through schema information or the data, or even create new database objects.

The Runtime Manager lets you create job schedules.

Unicode support

All components are designed to process and support virtually any representation of data. Unicode-enabled transformation functions can be used in calculations, scripts, and procedures.

Expressions

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

CHAPTER 2

Sybase ETL SBE Desktop

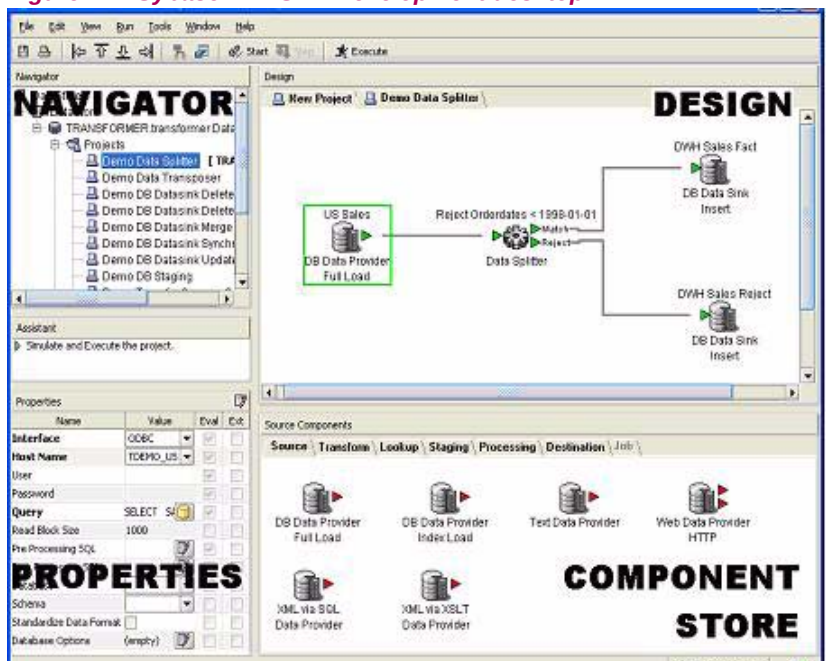
Desktop layout

The Sybase ETL Small Business Edition desktop consists of the following major sections:

- The Navigator section lets you select the repositories, projects or jobs that you want to work on.
- The Design section lets you design the project and job flow, drag components onto it, connect them with each other, and to simulate the data flow between them.
- The Properties section lets you set properties of the component currently selected in the Design section.
- The Component Store contains all components that are installed and available for designing projects. To add a component to the Design section, you can double click it, drag it to the Design section, or right-click and select Add from the pop-up menu.

Figure 2-1 shows an example of the Sybase ETL Small Business Edition desktop.

Figure 2-1: Sybase ETL SBE Development desktop



Using the Navigator

The following objects and functions are available from the Navigator:

- Navigate and browse the repository
- Administer projects and jobs

You can access Navigator commands from the menu bar. You can also right-click to open the pop-up menu on any desktop item in the Navigator.

Navigating and browsing a repository

In the Navigator, the hierarchical tree list represents the repository, the session to the repository, and objects stored in the repository, such as projects, jobs, and templates.

The following example shows the tree structure:

```
Repositories
-- <RepositoryName>
---- <ClientUser>.<Client>.<Repository Name>
------ Projects
-------- Project_1
-------- Project_2
-------- Project_N
------ Jobs
-------- Job_1
-------- Job_2
-------- Job_M
------ Templates
-------- Template_1
-------- Template_L
```

Administering projects and jobs

From the Navigator, you can administer projects and jobs. See Chapter 4, “Projects and Jobs” for detailed information.

Using the Properties section

The Properties section contains information and functions that let you:

- Review and modify all property items of the selected component
- Identify mandatory items
- Add component variables
- Allow dynamic evaluation of property items
- Encrypt property items

Whenever you select a component in the Design section, the property settings of the selected component appear in the Properties section.

A property name displayed in bold 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.

Refer to Chapter 6, “Components,” for a detailed description of properties of each component.

Applying component variables

You can add custom properties to components. Those parameters incorporate a variable that can be referenced inside user-defined procedures.

Adding a component variable

❖ **To add a component variable**

1 Right-click in the Properties section to open a pop-up menu.

2 Select Add.

3 Enter a name for the variable (for example, MyParameter). Inside the component, use the following notation to reference the variable:
`REF.MyParameter`

4 Enter a prompt and a description.

Editing a component variable

❖ **To edit a component variable**

1 Click the line containing the variable.

2 Right-click in the Properties section to open a pop-up menu.

3 Select Edit and enter the new settings.

Removing a component variable

❖ **To remove a component variable**

1 Click the line containing the variable.

2 Right-click in the Properties section to open a pop-up menu.

3 Select Remove.

Allowing dynamic expressions

Select the Evaluate option if you want 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.

Some property items are pre-selected as Eval. Check boxes indicate the current settings.

❖ **To enable or to disable evaluation for a property**

1 Right-click the property that you want to evaluate at runtime.

2 Select Evaluate to enable or disable evaluation.

Encrypting properties

Project and job data, as well as property values, are stored in the Sybase ETL SBE repository. Most of the records in the Sybase ETL SBE repository are not encrypted but are represented in a readable XML format.

❖ **To encrypt property values**

1 Right-click a property value.

2 Click Encrypt.

Refer to Chapter 6, “Components,” to find out about component specific property settings.

Using the Design section

In the Design section you can:

- Create and modify projects and jobs
- Simulate and run projects
- Run jobs

To create a project or job, you must add and connect components and then set the component properties.

Adding components

To add a component, select the component in the Component Store and drag it to the Design section. You can also right-click the component and select Add, or double-click the component.

Deleting components from the Design section

To delete a component, right-click it in the Design section and select Delete from the pop-up menu.

Using the pop-up menu to process commands

Right-click anywhere in the Design section to open the pop-up menu. When you right-click an empty area, the general project pop-up menu opens. The general project pop-up menu displays general commands, such as Close and Print. Right-click a component to open the Component pop-up menu. The Component pop-up menu displays component-specific commands.

Using the Component Store

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

❖ **To add a component to the Design section**

1 Select a component.

2 Add the component to the Design section. To do this, you can:

- Drag and drop the component to an empty spot or to an existing connection in the Design section.
- Right-click a component and select Add.
- Double click a component.
- Right-click a component to connect with a new component, select Add Right Component or Add Left Component, and select a component to add it to.

Customizing preferences

Use the Preferences window to customize the following groups of settings in the Sybase ETL Small Business Edition environment:

- Workbench
- Engine
- GRID Engine
- Performance Log

❖ To customize preferences

1 Select File | Preferences. The Preferences window appears.

2 From the Workbench list, select Appearance and set the following options:

- Locale to be used for the user interface – 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 – specify whether to show the The Assistant, which contains information regarding the current state of the open project. The Assistant helps guide you through the process of completing a project. This option is selected by default.
- Font for displaying source data – select the font that you want to use for displaying source data. This setting is useful when you work with non-western character sets, such as UNICODE character sets. The default font is Tahoma.
- Create new project on start-up – specify whether to start a new project automatically each time you start Sybase ETL Small Business Edition. This option is selected by default.
- Create new GRID projects – Not available.
- Create automatic link when components are added – specify whether to create a link to an existing component automatically when dragging a new component onto an empty spot within the project Design section. This option is selected by default.
- Display qualified transformation objects – specify whether the names of projects and jobs in the Navigator section are displayed, including the owner. This option is not selected by default.

- Use unique object name – specify whether to enforce unique project and job names on a repository connection. This option is not selected by default.

3 Select Query Designer and set the following options:

- Enable delete functionality of database objects – specify whether the Truncate Object command on the right-click pop-up menu can be selected to delete all records of a selected table. This option is not selected by default.
- Default amount of records to retrieve from Query Designe – specify the number of data records retrieved by the Query Designer. The default is 25.
- Auto Join Generation – specify whether to create joins automatically based on attribute names of the tables. This option is not selected by default.
- Use brackets in join generation – specify whether to use brackets when creating joins. This option is selected by default.

4 Select Engine and set the following options:

- Defaults to local engine operation – specify whether the engine restarts when you start Sybase ETL Small Business Edition. This option is selected by default.
- Engine Monitor update delay (sec) – specify the number of seconds to wait between two updates of the Engine Monitor. The default is five seconds.
- Simulation trace delay (milliseconds) – 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.

5 Select Grid Engine and set the following options:

- Grid Engine Server – specify the IP address of the primary GRID engine server.
- Grid Engine Port – specify the port address of the primary GRID engine server.
- Grid Default Port – specify the default port address of the primary GRID engine server. The default is 5124.

- Grid Engine Ping Timeout (sec) – specify the amount of time (in seconds) allowed for accessing the GRID engine before restarting. The default is 60 seconds.
- Progress Monitor update delay (sec) – specify the number of seconds to wait between two updates of the Progress Monitor for a job execution. The default is five seconds.

6 Select Performance Log and specify the detail level for logging performance data. The choices are:

- 0 = no log
- 1 = default level

7 Click Save.

A message may appear that indicates some of the changes require you to restart Sybase ETL Small Business Edition. To restart Sybase ETL Small Business Edition, click Yes. Click No to continue working. The changes take effect the next time you start Sybase ETL Small Business Edition.

CHAPTER 3

Getting Started

Starting Sybase ETL SBE

❖ **To start Sybase ETL Small Business Edition**

1 Double-click the Sybase ETL SBE icon or select it from the Sybase product group on the Windows Start menu.

By default, the "Welcome to Sybase ETL Development" page appears. It provides information that explains Sybase ETL Small Business Edition projects and jobs. To disable the page, clear the Show on Startup check box.

2 Click Close.

The Sybase ETL Small Business Edition desktop appears.

The Sybase ETL Small Business Edition desktop consists of the Navigator, Properties, Design Section, and Components Store. See "Desktop layout" for more information.

3 In the Navigator, click the repository folder and select the TRANSFORMER client to open the list of available projects.

Note When you open the project list, it displays the demo projects shipped with the product. Every demo project contains an example of how to use a component or how to implement a scenario.

4 Select an existing project, or right-click on Projects to create a new project.

Creating your first project

This section describes how to create and simulate a sample project with sample components. It does not explain all of the components, nor does it explain their properties and features. For details about components, see Chapter 6, "Components."

A project usually contains one or more of the following:

- Data providers that provide the data feeding the project data stream
- Data transformers that transform or remap field values

- Data sinks that write the transformed values to their target

Note The results of this section can be viewed in the "Demo Getting Started" project included as a demo project within your default repository.

Adding a data provider

Use one of the following methods to add a data provider to your project:

- Drag the component from the Component Store to the Design section.
- Right-click the component that you want to add and select Add from the pop-up menu.
- Double-click the component.

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

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

❖ **To configure a data provider**

1 Select ODBC from the Interface drop-down list. (See "Entering database connection parameters" on page 68 for information about all of the Interface types.)

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

After you confirm the initial component settings, the settings appear in the Properties section.

3 To define what information should be retrieved from the data source, click Edit on the Query property.

The Query window appears.

4 Enter a SQL Query or click Query Designer to generate the necessary SQL.

The left section 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 to the Design section, or right-click the table name and select Add Object to Query.

6 Click the PRODUCTS table and drag it to the Design section.

7 Click Save to close the Query Designer. You return to the Query window. The SELECT query has been generated automatically.

8 Click Execute the Query to run or test the query. You can also modify and edit 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

❖ **To add a data sink**

1 In the Component Store, go to the Destination tab and select the DB Data Sink Insert component by dragging it into the project.

2 Select ODBC from the Interface drop-down list.

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

4 Click the selection button of the Destination Table property and select PRODUCTS from the table catalog.

5 Click Finish to confirm your settings.

Your project should now consist of two components. The link between the components had been created automatically (provided the setting Create Automatic Link When Components Are Added has been activated in your Sybase ETL Small Business Edition preferences). If the line has not been automatically created, you can easily draw it by clicking on the output port and dragging it onto the input port of the Data Sink.

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 are both displayed in green. This indicates that both components have been completely configured (in other words, all required information has been provided).

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

❖ **To review and set properties for a component**

1 Right-click the connecting line (the Link) between the components. The line changes to the color green and a pop-up menu appears.

2 Select the Mapping command.

The mapping between the data source and the target source has been 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, you can easily free a target connection point by selecting and deleting the mapping line that is currently linking to it.

Adding a data calculator

❖ **To add a data calculator**

1 Click the Transform tab in the Component Store.

2 Select the Data Calculator Java Script component and drop it 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 to the left.
- The Data Calculator window appears.

The Data Calculator window has a Tabular and Graph view:

- Use the **Tabular** view to enter transformation rules.
- Use the **Graph** view to visually define the mapping sequence between the input port and the output port.

3 Click the Graph tab. Two sections IN and OUT represent the current structure of the port attributes.

You are prompted to assign a default mapping by order.

4 Click Yes.

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 From the File Menu, select Save to save your project.

Simulating the project

❖ **To start the simulation**

1 Click Start in the second-level 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 from the pop-up menu to preview data on the link.

CHAPTER 4

Projects and Jobs

Managing projects

Projects are the working units of Sybase ETL Small Business Edition. A project 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 ETL Small Business Edition project starts with one or multiple source components and ends with one or more destination components.

The following list summarizes the 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 type.
- If a Transformation component allows multiple input data streams, multiple originating **Source components** are required.
- If a Transformation component has more than a single output of data streams, each data stream can be connected with a component.

Creating a project

❖ **To create a project**

1. Select New | Project from the File menu. You can also right-click a project, job, or template in the Navigator and select New | Project.
2. Drag components from the Component Store onto the Design section as your project requires it.

Executing a project

To execute a project, right-click the project in the repository and select Execute from the pop-up menu.

Modifying a project

To modify a project, double-click the project name in the Navigator section. The project opens and you can make changes.

Unlocking a project

If a project is locked, the project can generally be opened in read-only mode. To make a project available for read/write access, select Unlock.

Copying a project or job

To copy a project or job, you can select it from the Navigator tree, right-click, and select Save As.

If you select Save As, you create a copy of an existing project or job, leave the original untouched, and store no reference to the originating project or job.

Deleting a project

The Delete command deletes a project from a repository. To delete a project, right-click it in the Navigator section and select Delete from the pop-up menu.

Renaming a project

Use the Rename command to rename the current project.

Resetting execution properties

Resets the current value of the Load Index Value (DB Index Load component). This is used to reset loading options for incremental load.

Simulating and executing a project

Simulating a project is a highly interactive process that lets you monitor and validate your transformation process step by step. In contrast to executing a project, you can view the data during a simulation 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 on the fly to validate your rule base for all potential content.

Note A project can be simulated only after all components have been properly initialized.

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

- Start a simulation
- Step through a component
- View the data flow on the connecting Link or within the component
- Modify and re-initialize 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 data and output data inside a component
- Modify properties or calculations on-the-fly, so that you can change transformation rules and sample values to validate your rule base
- Re-step a component after modifying a calculation or property
- Perform “what-if” scenarios

- Take multiple steps through the project

❖ **To simulate a project**

1 Click Start to start a simulation. When you click Start:

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

2 Select a component and click Step to execute the component.

Stepping a component means to execute or process a single component with the data that is currently available at its input ports. The data records that are being processed during a single step are the records currently populating the IN-port 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 the transformation process, you can examine the link between components or the ports of a component. Other components like Data Calculator and the Data Splitter include built-in preview capabilities.
- To view data on the link, right-click the connecting link and select Preview from the pop-up menu.
- To view data currently at the port, right-click the port and select Preview from the pop-up menu.
- To view data from inside the component, double-click the component, or click Rule in the Property section.

  Use this option to see the impact of transformation rules from within components, like Data Calculator or the Data Splitter.

4 Modify and initialize the component.

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

To modify and initialize a component:

a Select the component and modify its properties.

b Save the modifications.

c Right-click the component and select Initialize from the pop-up menu.

Tracing project execution

To simulate a project step by step without interaction, choose Trace. Trace steps through the project, highlights the current component, and updates connection record counts.

If the simulation has not been started, Trace starts it automatically, and continues until all data has been processed. Click Stop in the Trace working message to interrupt the simulation.

Note Tracing does not perform post-processing like executing Post SQL on DB Components, etc.

You can control the simulation rate by setting Simulation trace delay option. The Simulation trace delay option accepts values between 10 and 9999 milliseconds, the default is 250 milliseconds.

❖ **To set the trace delay**

1 Click File | Preferences | Engine.

2 Change the Simulation trace delay.

Simulating the entire project

To run the entire project in the simulation engine, choose Simulate.

Note Simulation can not be cancelled.

Executing the current project in default grid engine

To execute the current project in the default grid engine, choose Execute. As the project definition is read from the repository, you must save changes in the project before execution. If you do not save the changes, execution will not start. Execution has no impact on the state of the simulation.

Viewing current mappings

The Mapping Definition window shows the current mapping between attributes of the adjacent IN- and OUT- structures. To open the Mapping Definition window, right-click the connecting link and select Mappings from the pop-up menu. Selecting Display Structure and values from the list box in the toolbar displays the fields as well as the current values.

Note 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

From the Mapping Definition toolbar, you can automatically create mappings by selecting Create Mapping By Order or Create Mapping By Name.

If you select Create Mapping By Order, the port attributes of the IN- and OUT- structures are mapped sequentially.

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

If you select Create Mapping By Name, the port attributes of the IN- and OUT- structures are mapped according their names.

Applying manual mappings

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

You can also change existing mapping lines. If you want to change the current mapping, select the mapping line at the connection point and drag the line to an unmapped port attribute.

To delete a single mapping, select the mapping line and press the Delete key, or right-click and select Delete from the pop-up menu. To delete all mappings of a link, click Delete, or select Mapping | Remove All.

Managing port attributes

The Structure Viewer is available at the port of a component. You can add and delete port attributes, or modify the settings or existing attributes.

To open the Structure Viewer, right-click the port and select Edit Structure from the pop-up menu.

❖ **To add an attribute to the port structure**

1. Click the line number of the line where you want to insert the attribute. The current and following attributes will shift downward.
2. Click Add Attribute, or select Actions | Add.
3. Enter a name for the attribute. The names for port attributes must start with an alpha character and can contain only alphanumeric characters (A-Z, 0-9).
4. Enter a datatype.
5. Enter a size.
6. Enter additional specifications.
7. To delete an attribute from the current port structure, select an attribute line number and click Delete, or select Actions | Remove.

Modifying datatypes

When you modify datatypes of a record structure, you modify the internal logical representation that Sybase ETL Small Business Edition uses for the record structure during your transformation. This does not going change the data structure definition of the source or destination tables. You must make sure that the data structure of the final Data Sink is compatible with the content you are generating. If you are using the Standardize Data Format option, make sure that the datatype you are assigning to the port attribute matches the type of your target attribute.

Viewing a simulation flow

After a simulation has been started, the flow of the simulation is made visible through:

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

The number of records being processed within each single step is dependent 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 while a project executes.

Stepping from current and selected component

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

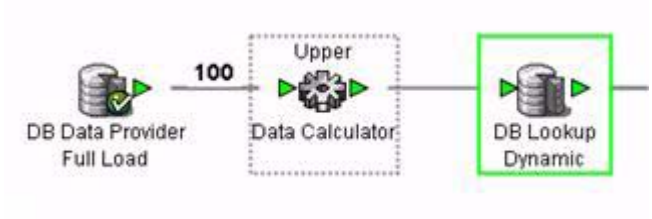
When you step through the project without making modifications, the box moves from component to component, eventually reaching the Data Sinks. Only the Data Sink components do not cause the box to move, if they are stepped. Instead of moving the active box, they display success or failure icons.

As you move, the dotted box highlights one component to another current component.

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

The following example shows the Data Calculator as current and the DB Lookup Dynamic as selected.

Figure 4-1: Example of current and selected components



The current component is the component that will be executed next when you select the Step button in the toolbar (or Simulate | Step). If during the simulation process, you want to inspect or change a component that is different from the current component, then click it. The solid green box highlights the selected component. You can resume the simulation after a component property changes from either the selected or the current component.

To resume simulation from the selected component, select Step from the pop-up menu. To resume simulation from the current component, select Step from the toolbar.

Note Selecting Step from the pop-up menu for a component that has not yet been processed forces all previous components to be stepped first.

Forwarding and backward-forwarding components

The visible flow of the simulation as indicated by the box is straightforward in many projects. The box moves from one component to the next. However, the flow of a project simulation does not necessarily head in one direction. The flow of simulation heavily depends on the components used within the project.

There are forwarding components, which receive a number of records, apply the transformation to those records, and forward those records (such as the Data Calculator and the Character Mapper). The number of records processed in a single step is determined exclusively by the value of the Read Block Size property of the preceding component.

Other components do override the previous Read Block Size parameter. 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 has been delivered to the IN-port. The Staging component resizes the number of records forwarded with the next step by using its own Read Block Size property. See Chapter 6, "Components" for an explanation of the behavior of every component during the simulation.

Previewing data from multiple locations

The Preview command is available at every connecting link, every port, and every component. The Preview command opens the Preview Content Browser to display the data currently available at the selected location.

The Preview Content Browser window includes tabs that allow you to display multiple previews from multiple locations simultaneously. Sometimes, it is extremely useful to preview the content of the IN-port and OUT-port of a component in parallel tabs.

Partial execution or initialization during simulation

It is extremely time consuming to start the entire simulation after making modifications to a single component, especially when working with a large number of input records in a project that consists of dozens of components. On the other hand, it can 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. The commands Step through and Start through provide an effective way of 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 Start Through from the Simulate menu. The simulation starts the current project, processes all components between the current and the selected component, and finally 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. You 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 like the Staging component offer to customize values for both reading and writing.

Note The Block Size property is evaluated during both project simulation and project execution. A small number might be suitable for simulation purposes, but will slow down execution time when you click Execute. In simulation the Block Size is restricted to 32K.

Controlling multiple data streams

While most projects will consist of a single stream of components connected through links, it is technically possible to have multiple, not interconnected data streams within a single project. Since Sybase ETL is a parallel system, there is no way to predict in which order the streams are processed.

If you have multiple data streams it is highly advisable to design a project for each data stream so that all components within a project are connected to each other. By following this design guideline, it is easy control the data streams by connecting the projects to form a job process flow.

Managing jobs

A **job** lets you easily set up the powerful control flows for one or multiple projects. Although Sybase ETL Small Business Edition projects require some kind of user interaction during simulation or execution, a Sybase ETL Small Business Edition job can be scheduled to run without any user interaction.

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

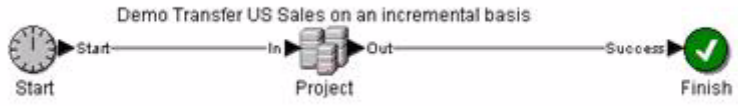
Job components

A job consists of at least one of the following components:

- A Start component
- A Project component
- A Success component

Figure 4-2 shows an example of a job with one Start component, one Project component, and one Success component.

Figure 4-2: A Job with the minimum components

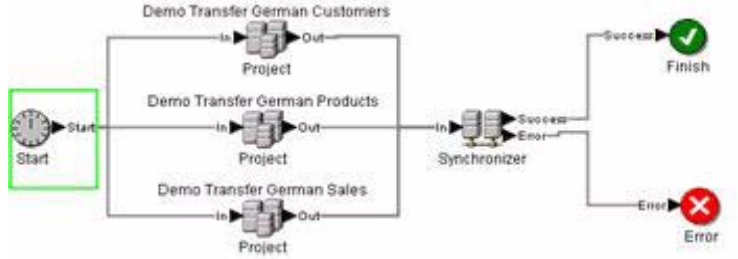


A job can be extended to include:

- Multiple projects in sequential or parallel order
- Multiple synchronizers
- Multiple Success and Error components

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

Figure 4-3: A Job with multiple components



Executing a job

A job can be executed directly from the desktop or at specific time intervals as a scheduled task of the operating system Task Manager. To schedule a job, click Start Time in the Property section of the Job Start component, or select Runtime Manager from the Tools menu.

Creating Jobs

To create a job, right-click the Jobs entry in the Navigator section and select New | Job from the pop-up menu. The available job components are displayed in the Component Store.

❖ **To create a job with minimum components**

1. Add the Start component from the Component Store to the Design section.
2. Add the Project component and connect it to the Start component.
3. Add the Finish component and connect it to the Project component.
4. Double-click on the Project component to open the project browser window.
5. Select the project you want to make part of this job.
6. Close and save the job. The job is now ready to be executed in the Sybase ETL Small Business Edition desktop or as a scheduled task.

From the Navigator section, you can directly display and access the projects included in a job by opening the sub-branch.

Modifying a job

Double-click the job name in the Navigator section or select Open from the pop-up menu.

Copying a job

To copy a job to a specified destination, open the job and select Save As from the File menu.

Deleting a job

❖ **To delete a job**

1. Right-click the job in the Navigator.
2. Select Delete from the pop-up menu. The Confirm Job Delete dialog box appears.

3 Click Delete. By default, only the job is deleted. You can also delete the job and all included projects by selecting the Delete Included Projects check box and then clicking Delete.

Note When using this option, be sure that related projects are not used in other jobs as well, because this is not checked automatically. Projects that are currently open for design (locked) are not affected.

Renaming a job

Highlight the job listed in the Navigation tree, right-click, and select Rename.

Scheduling a job

❖ **To send a job to the Task Scheduler**

1 Open the job.

2 Click the Start component.

3 In the Property section, click Edit Schedule Job of the Start Time property.

4 Enter the task parameters as described in the Runtime Manager section.

Controlling job execution

You can control Job execution as follows:

By using a

- synchronizer component that allows you to branch job execution based on a project's success or failure
- By ignoring errors on each project

Refer to "Job components" on page 141 for more detail about job configuration.

Using templates to create projects and jobs

Templates provide a way to automatically create projects and jobs for special purposes. Currently only Migration templates are available.

Building a migration template

❖ **To build a migration template**

1 Right-click the Templates entry in the Navigator section, and select New | Template from the pop-up menu. The Template Assistant guides you through the steps necessary to build a fully configured migration template.

To modify an existing template, select Open from the pop-up menu in the Navigator section.

2 Follow the Template Assistant by first entering the details for the job to be created. The Name value is used for the template object and, further qualified, for the generated transformation objects.

The currently available Migration Type is DB to DB.

3 Provide information about the source database and select the tables to transfer:

- Database connection properties – The database connection properties are the same as for the DB components. The Advanced option lets you set special database options. See “Entering database connection parameters” on page 68 for more information.
- Table properties – To get the table catalog for the specified database, select Logon. The available tables are listed in the lower section of the window. By default, each table is selected for transfer. To exclude tables you can directly deactivate the Transfer check box or select multiple tables and choose Exclude from the pop-up menu.

  You can view additional information about the tables by clicking Browse or Count on the pop-up menu. The Count All option displays the record count for all tables.

4 Enter database connection properties for the destination database.

5 Enter transfer settings.

Additional properties for each table to be transferred are provided.

a Select source attributes.

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

Deactivate the Transfer check box for every attribute you want to exclude from transfer, or select multiple attributes and choose Exclude from the pop-up menu.

b Select destination tables.

It is assumed that source and destination table names are equal. To use different names, you must provide these for each source table by either entering a new name into the Destination field or selecting an existing table from the drop-down list.

c There are some additional options that can be switched on or off. To change the value of an option you can either click a single check box or select all lines you want to change and choose Activate or Deactivate from the pop-up menu:

- Data model options – Before the transfer can start, you must verify that the destination tables exist. These options can help you set up the destination data model. They do not affect execution but they do 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. If you want a table to be re-created even if it exists, select Drop Table.

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

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

  The failure of a Critical project causes the job to stop execution and signal failure. This and the Ignore Errors options correspond to the properties of the Multi-Project job component. The Ignore Errors setting does not affect the projects generated through this template.

6 Process migration template data.

The last screen in the wizard allows you to perform all desired tasks on the collected data.

Except for Save, you can perform all tasks from the template pop-up menu in the Navigator section:

- Storing the Template – If you select Save Template, the template is stored in the repository. For a stored template, you perform all other tasks from the pop-up menu in the Navigator section. Storing allows you to reuse the collected data for similar jobs.
- Generating Transformation Objects – Select Build Transformations to create one project for each source table and a migration job that controls the execution of all these projects.
- Creating the Destination Data Model – To set up the destination data model according to the data model options you entered, select the Create Data Model.
- Executing the job – The Execute Job option is available only if Build Transformations is activated. If you select this option after the migration template data has been processed, the generated job is executed.

7 Perform selected tasks.

To finish collecting data and perform the selected tasks, click Finish.

Note Be sure to select at least Save template or Build Transformations. Otherwise the collected data will be lost.

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

Managing a migration template

Creating a template

To create a template, right-click the Templates entry in the Navigator section and select New | Template from the pop-up menu. The Template Assistant guides you through the necessary steps to a fully configured migration template.

Modifying a template

Double-click the template name in the Navigator section or select Open from the pop-up menu. The Template Assistant guides you through the necessary steps to a fully configured migration template.

Copying a template

Select Copy from the pop-up menu and enter a name for the new template.

Deleting a template

Select Delete from the pop-up menu to remove a template from the repository.

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

Renaming a template

Select Rename from the pop-up menu 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, select the Build command. To enforce unique names, a creation timestamp is added to all object names.

Note You can modify the generated jobs and projects before execution like any other object you created manually.

Creating a data model from a template

To set up the destination data model according to the data model options stored with the template, select Create Data Model from the pop-up menu.

CHAPTER 5

Advanced Concepts and Tools

Content Explorer

Content Explorer consists of a menu, a toolbar, a SQL structure area, a Navigation area, and the Design area. From the Content Explorer window, you can customize the size of the areas. You can also control the structure of the SELECT area by using the commands available in the Options menu.

Use Content Explorer to do the following:

- Browse the table catalog of any connected database of the current project in the Design section
- Easily create SQL queries by using a powerful graphical user interface
- Review the generated SQL statement
- Execute SQL queries against the database
- Browse the data of a selected table or view
- Create a table in the schema
- Delete all records of a table
- Count the number of records in a table of view

The examples in this chapter open the project Demo Getting Started from the Demo Repository.

Opening Content Explorer

To open Content Explorer, select Content Explorer from the Tools menu. The Choose Data Source dialog box appears and lists all 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.

Double-click the line to open the Design window of the Content Explorer. The examples in this chapter are based on the DB Data Provider - Full Load component in the project Demo Getting Started from the Demo Repository.

In the Table Catalog on the left, you can directly browse the content of an object selecting Browse from the pop-up menu.

Using the Design area

The Design area is part of a graphical user interface that enables you to automatically generate SELECT statements that are more specific about the records you are interested in.

From the Design area, you can:

- Drag objects from the Navigation area into the Design area
- Select attributes to be used in the SELECT/WHERE/GROUP/ORDER BY clause
- Graphically create joins between tables using SQL 92 join operators

Creating queries

The Content Explorer allows you to create queries in a convenient graphical environment. Little or no SQL knowledge is required.

Note You can only use the Content Explorer to generate ad hoc queries, which cannot be saved to a file or to the repository. However, if you want to save the generated SQL for other purposes, you can copy it from the Generated Query window that can be opened by selecting Generated Query from the View menu. To do so, select the part of the generated query and copy it to the clipboard.

❖ To create a simple query

The following procedure uses the PRODUCTS table to generate a simple query that retrieves all attributes from the a table.

1. Click the table or view name in the Navigation area.
2. Drag the selected object to the Design area.
3. Verify the results of the generated query by clicking View the Generated Query.

❖ To create a query using multiple tables

The following procedure uses the PRODUCTS and the SALES table to generate a query that retrieves joined information from two tables.

1. Click the table PRODUCTS and drag it onto the Design area.
2. Click the table SALES and drag it onto the Design area.

3 Create a join between the tables by drawing a link between the fields PR_ID of both tables. The join can also be automatically created if you select the Auto Join Generation option in the Preferences.

❖ To use the Auto Join Generation option

Auto Join Generation is based on identical attribute names used within tables or views. If there are identical names and the Auto Join Generation option is selected, the Query Designer automatically creates a join based on those attributes.

1 Access the Preferences dialog box by selecting Preferences from the File menu.

2 Select the Workbench | Query Designer | Auto Join Generation check box in the Preferences dialog box.

❖ To modify the default setting 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. The default is Equi Join.

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

2 Select the Modify command.

3 Choose a join type from the list:

- = EQUI Join
- += Left outer Join
- =+ Right outer Join
- +=+ Full outer Join

The default setting of the join changes to the type you selected.

❖ To add one attribute to the SELECT clause

1 Drag the tables to the Design area (if they are not there already).

2 Click the attribute names you want to add.

3 Right-click and select Add Items to Selection.

❖ To select more than one attribute to the SELECT clause

1 Drag the tables to the design area (if they are not there already).

2 Hold the Ctrl key and click the attributes that you want to add to the SELECT clause.

3 Right-click and select the Add Items to Selection.

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

1 Click the header (name) of the table in the Design area.

2 Right-click and select Add Items to Select.

❖ **To view generated SQL statements**

- To display the statement as currently generated by the Content Explorer, click Generated Query or select Generated Query from the View menu.

❖ **To add functions to the SELECT attributes**

1 Click the attribute and right-click to open the pop-up menu.

2 Select any of the available functions.

Inspecting log file information

The File Log Inspector window allows you to inspect log file information about job execution, fatal errors, and the system log. The log files are located in the *\log* subdirectory of the installation directory. The log files are:

- *execution.log* — captures all information regarding job execution errors.
- *fatal.log* — captures low-level information that is written when the system encounters serious unexpected behavior. This includes information from fatal system exceptions when the system was no longer able to write to the *system* log file.
- *system.log* — captures all information about system activities, both operational and exceptional events. The detail of data written to this file depends on the Trace Level that is set in the *default.ini* file located in the */etc* directory. You can also change the Trace Level within a project or job by using the uTracelevel(n) function in a JavaScript procedure.

A Trace Level of 0 (the default) only traces minimal processing information. A Trace Level of 5 traces the maximum amount of information about processing events and processing steps.

Note Even a Trace Level of 1 will considerably increase the amount of logging information written to the *system.log* file and, as a result, will impact the overall performance of the Sybase ETL SBE environment.

The function uTracelevel(*n*), where *n* is a value of 0 through 5, lets you set the Trace Level from within a project or job. As a result, you can trace the execution of a single component with maximum detail, while all other components are only tracing at the default level. You can call the uTracelevel function from within a JavaScript procedure.

Managing jobs and scheduled tasks

The Runtime Manager manages jobs and gives you an overview of your currently scheduled job tasks. Using Runtime Manager, you can create, edit, delete, execute, and terminate tasks.

Because Runtime Manager is based on Windows task scheduling manager, you will find any scheduled task currently defined on the system.

To open Runtime Manager, select Tools | Runtime Manager.

❖ To create a new schedule

1. Create a new task by clicking Create a New Schedule on the toolbar or select Actions | Create. The New Schedule window is displayed.
2. Select a Job in the New Schedule window.
3. Edit Name and Description.
4. Enter Username and Password for the Windows user account that will run the job.

Note Because this is a separate task, Windows expects a valid username and password. A password is required for the account. The user must have read/write access to the ETL user folders, which can be the installation directory or the Windows user directory, depending upon the type of installation.

5 Click Create to confirm your settings.

❖ **To execute a job schedule**

1 Select the scheduled job from the list.

2 Click Execute a Schedule, or select Actions | Execute.

❖ **To delete a job schedule**

1 Select the scheduled job from the list.

2 Click Delete a Schedule, or select Actions | Delete.

❖ **To edit a job schedule**

1 Select the scheduled job from the list.

2 Click Edit a Schedule, or select Actions | Edit.

❖ **To terminate a job schedule**

1 Select the scheduled job from the list.

2 Click Terminate a Schedule, or select Actions | Terminate.

After it is scheduled, the job is executed under control of the Windows Task Scheduler. In the Last Result column of the Task Scheduler, you can find the execution state of the job.

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

Table 5-1: Job execution state codes

Result	Type	Description
0	Info	Job execution successful
1	Warning	Job execution cancelled
101	Warning	No valid license
10001	Error	Unable to retrieve job data from repository
10002	Error	Unable to find initialization component for job
10003	Error	Unable to initialize external job settings
10004	Error	Initialization failed
10005	Error	Job execution failed
10101	Error	Connect to repository database failed
10102	Error	No valid repository found in connected database
10103	Error	Unable to open session on connected repository

Customizing SQL and transformation rules

Setting up a project or job includes a variety of tasks, some of which can be customized:

- Entering SQL queries to set up the source components
- Entering SQL commands for pre-processing and post-processing tasks
- Entering expressions, conditions, and procedures to manipulate the transformation process

Although the format of the SQL commands is strongly dependent on the database system that is connected to the component, the format of the transformation language supported by Sybase ETL SBE (JavaScript) will not change, no matter what source or target system you are using in your projects.

The JavaScript expressions can be included in Square Bracket Notations (SBN expressions), which can considerably reduce your customization efforts. SBN expressions can be part of component properties (if the Evaluate property is activated for the specific property), SQL statements, or any pre- or post processing commands. An SBN expression resides inside an opening and a closing square bracket and is evaluated at “Runtime,” as opposed to constant expressions that are defined at “Design Time.”

Using 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. 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. A procedure can be written in JavaScript.

Examples of procedures are:

```
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';
}
```

Including 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.

There 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 or OUT. for the OUT-Ports. IN-Port variables are read-only, OUT-Port variables can be written. The following example uses PORT variables in an expression:

```
uUpper(IN.CU_NAME)

Using PORT variables in a procedure:

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

Component variables

Component variables are created in the Property section of the component and can be referenced inside the component. The Component Variable is only valid inside the component. The name of the variable is prefixed by REF, for example:

```
uIsNull(REF.myvariable)
```

Note To provide high flexibility in transformations, all port and component variables internally use the datatype “string”. This may result in unexpected behavior when using numeric values. If multiplied by 1, the numeric value of a string variable will be 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
```

Using functions

Sybase ETL SBE provides a complete set of functions and operators based on a design that integrates complete support for the Unicode character sets.

Sybase ETL SBE functions can be recognized by the prefix u, for example, uConcat().

Using 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.

SBN expressions can be used 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.

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

The following expression is used to specify the path of a file in the Text Data Provider:

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

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

Entering SQL statements

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

To enter a query for the component, select the Query option of the Query property.

From the Query window you can:

- Enter a query
- Run a query
- Save a query
- Open the Query Designer
- Look up the database schema

Entering queries

You can manually enter a SELECT statement into the Query field. You can use any valid SQL notation of the connected database to build the query.

To open the Query Designer, click Query Designer.

To look up the database schema, click Lookup Schema.

To run the query, click Execute Query.

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

Validating queries

The query is immediately validated against the database system that is connected to the component. Therefore, the query syntax must be compliant with the native SQL dialect the connected database system is using. Using SQL 92 ANSI Standard queries allows switching to different database systems without changing the SELECT statements.

Query designer

The Query Designer is embedded in the Content Explorer, which is explained in detail “Content Explorer” on page 42.

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 component variable. Assuming that value '12345678' was assigned to CustNo, the SELECT statement with the dynamic expression would look like the following example:

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

You can use any of the Sybase ETL SBE functions inside the SBN. The following statement returns the same record using a value of *"1234"* for CustNo1 and a value of *"4567"* for CustNo2:

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

Manipulating the TRACE level at the start of a query

To manipulate the TRACE level at the start of a query, `SELECT * FROM PRODUCT [uTracelevel(5)]`.

The function uTracelevel() returns no value; therefore, the SBN expression [uTracelevel(5)] can coexist with the SQL statement. After the SBN expression has been evaluated, the SQL statement to execute will be:

```
SELECT * FROM PRODUCT
```

You can even execute the following query:

```
[uTracelevel(0)]
```

Pre-processing and post-processing SQL

For any component with database connectivity you can enter pre-processing and post-processing SQL statements in the Property section. Those properties allow entering SQL statements that are executed during initialization (pre-processing) or after completion (post-processing) of the component.

Some considerations are:

- The SQL statements will not return output after being executed.
- Any SQL statement accepted by the connected database system is allowed.
- You can enter multiple SQL statement in the pre-processing or post-processing SQL property by using a semicolon; as a statement delimiter.
- SBN expressions are allowed in pre-processing and post-processing SQL.

The following examples show pre-processing and post-processing SQL:

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

Using the JavaScript Procedure Editor and Debugger

JavaScript is an object-oriented scripting language designed for embedding into other products and applications. The language is divided into the core JavaScript and client-side JavaScript. The client-side JavaScript is designed to manipulate objects in Web browser, and the core language can be used in multi-purpose environments.

Inside the Sybase ETL Small Business Edition environment, core JavaScript is embedded to allow manipulation of objects to provide programmatic control over them.

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

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, the scripts entered can be executed and tested 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-definable Breakpoints to stop code execution at any line positions
- User-definable Go points to arbitrarily choose the position from which a code shall be executed
- Step mode to execute the code line by line
- Step-over during debugging
- Evaluation of JavaScript expressions
- Verify the result of code execution

Starting the JavaScript Editor and Debugger

Within the Data Calculator JavaScript component, click Edit to open the JavaScript Editor and Debugger.

The JavaScript Editor contains the following areas:

- Toolbar — consists of option buttons that you select to perform various JavaScript Editor functions. Refer to the online help for details about the toolbar.
- Object Navigator — consists of the Variables tab and the JavaScript tab. The Variable tab consists of input and output port variables, as well as temporary and pre-defined variables. From the JavaScript tab, you can access all functions, commands, and system variables that can be applied within the procedure.
- Edit/Debug — lets you edit the actual code. The area provides color-coding of syntax structures.
- Monitor — consists of the following tabs:
  - Tasks — contains the results of the validation after your procedure has been compiled.
  - Watch List — displays selected variables and their (changing) values while stepping through the code during debugging.
  - Input Records — displays the content of the current input record. To synchronize Input and Output Record, click Simulate in 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 clicking the Evaluate button.

Edit and Debug mode

When launched, the JavaScript Editor and Debugger comes up in Edit mode. To switch to Debug mode, you can:

- Select Start from the Debug menu.
- Click Compile.

A dark grey background of the edit area indicates Debug mode.To switch from Debug mode to Edit mode, click Start Editing.

Editing and debugging JavaScript

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

To validate JavaScript code, click Compile. The result of the validation is displayed in the Tasks tab of the Monitor area at the bottom area of the Procedure Editor.

The Editor offers some 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.

❖ To step through the code

Note The JavaScript Editor and Debugger will work without having input data at the input port of the component. However, in order to produce meaningful results, it is best to populate the input port with data before using the debugging features.

1. Before stepping through the script, either validate the script or switch to Debug mode.

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

2. Make sure, that the result message in the Task tab contains “successful compilation.”

3. To move to the next line, click Step.

   At any point during stepping you can inspect the current value of a variable. To do this, double-click the variable to select it, and then right-click to open the pop-up menu. The pop-up menu displays the variable name and the current value.

❖ To add and remove Breakpoints

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

1. To include Breakpoints, click on the line where you want to set the Breakpoint.

2. Right-click and select Add/Remove Breakpoint from the pop-up menu.

3. To remove a Breakpoint, right-click it and select Add/Remove from the pop-up menu.

❖ To step to a Breakpoint

1. Click Go for each step.

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

Inline inspection of variables

You can perform an inline inspection of the current value of a variable while stepping through the code in Debug mode or after the code has been executed. Right-click the variable to open the pop-up menu, which displays 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.

❖ To add a variable to the Watch List

1 Right-click the variable.

2 Select the Add to Watchlist from the pop-up menu.

❖ To remove a variable form the Watch List

1 Right-click the variable in the Watch List tab of the Monitor area.

2 Select Remove *<variable>* from Watchlist from the pop-up menu.

Special JavaScript features

Interrupting execution

From inside the Editor, click the Interrupt button to interrupt a JavaScript execution.

Creating user-defined errors

Using the throw("*xx*") function, an error can be enforced to interrupt the execution of the project. For example, 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

Functions can be defined inside a script and functions can be nested. For example, the following 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 the Sybase ETL SBE are represented as strings. This may result in unexpected behavior when working with numeric values. The functions parseInt() and parseFloat() can be used 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
```

Execution Monitor

Once a project or job is started from the Navigator, Sybase ETL displays the Execution Monitor. The Execution Monitor window displays properties about the current job/project.

Job properties

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

Property	Description
Name	The job/project name.
State	The current job execution state
Start	The start date and time.
Stop	The stop date and time.
Message	Error Message.

Projects

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

Property	Description
Name	The project name.
State	The current project execution state
Start	The start date and time.
Stop	The stop date and time.
Engine Name	The name of the executing engine.
Engine Host	The host of the executing engine.
Engine Port	The port of the executing engine.
Message	Error message.

Cancelling Job Execution

To cancel job execution:

- Click the Cancel Execution button.

GRID engines will try to cancel running projects. Projects still waiting for execution will not be started.

Analyzing performance data

While executing jobs and projects, Sybase ETL SBE collects performance-relevant data and stores it in a repository table. This section describes data model and content. It also provides some examples of analytic reports based on that data.

❖ **To collect performance data**

1. To collect and store performance relevant data on executing jobs and projects, select File | Preferences | Performance Log.
2. From the Logging Level list, select 1 to collect performance data.

Performance data model and content

The performance data is stored in a single, de-normalized, repository table named TRON_PERFORMANCE. 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.
process	prepare job	Perform job pre-processing.
process	prepare project	Perform pre-processing for a project within a job.
process	finish project	Perform pre-processing for a project within a job.
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.

Example reports

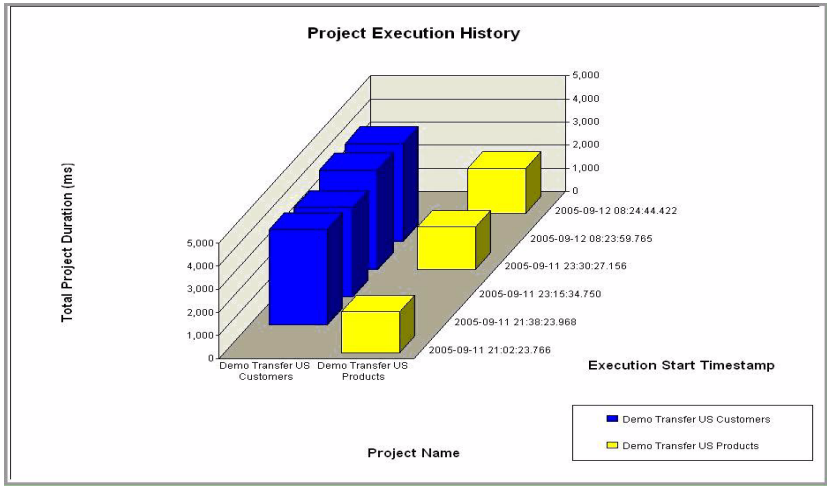
This section shows some examples for analytic reports based on the collected performance data.

Note You can use any appropriate analyzing tool for generating performance reports. Not every tool might offer the functionality to build reports similar to the examples shown.

If you want to generate your own reports based on the performance data, see “Performance data model and content” on page 60 for a description of data model and data.

Project execution history

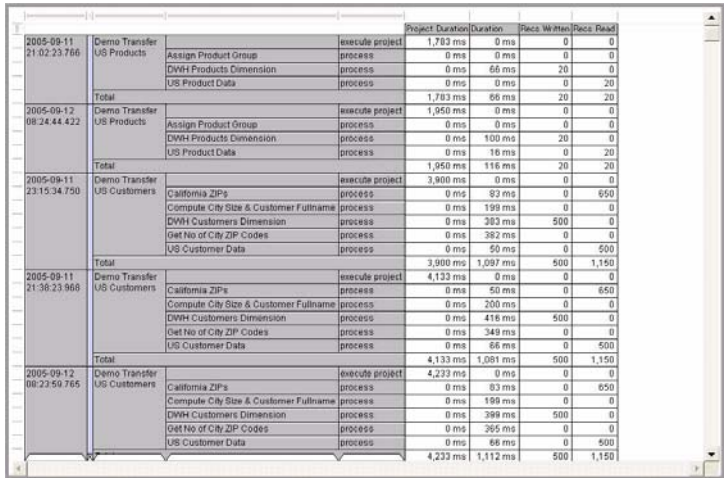
Figure 5-1: Example of a project execution history report



Project execution time and records moved

Tabular

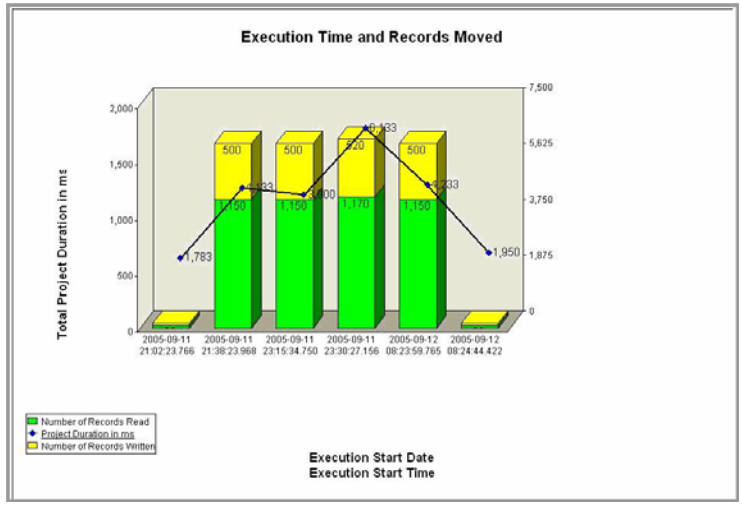
Figure 5-2: Example of a tabular report capturing project execution time and records moved



				Project Duration	Duration	Recs Written	Recs Read
2005-09-11 21:02:23.766	Demo Transfer US Products		execute project	1,783 ms	0 ms	0	0
		Assign Product Group	process	0 ms	0 ms	0	0
		DWH Products Dimension	process	0 ms	66 ms	20	0
		US Product Data	process	0 ms	0 ms	0	20
	Total			1,783 ms	66 ms	20	20
2005-09-12 08:24:44.422	Demo Transfer US Products		execute project	1,950 ms	0 ms	0	0
		Assign Product Group	process	0 ms	0 ms	0	0
		DWH Products Dimension	process	0 ms	100 ms	20	0
		US Product Data	process	0 ms	16 ms	0	20
	Total			1,950 ms	116 ms	20	20
2005-09-11 23:15:34.750	Demo Transfer US Customers		execute project	3,900 ms	0 ms	0	0
		California ZIPs	process	0 ms	83 ms	0	650
		Compute City Size & Customer Fullname	process	0 ms	199 ms	0	0
		DWH Customers Dimension	process	0 ms	383 ms	500	0
		Get No of City ZIP Codes	process	0 ms	382 ms	0	0
		US Customer Data	process	0 ms	50 ms	0	500
	Total			3,900 ms	1,097 ms	500	1,150
2005-09-11 21:38:23.968	Demo Transfer US Customers		execute project	4,133 ms	0 ms	0	0
		California ZIPs	process	0 ms	50 ms	0	650
		Compute City Size & Customer Fullname	process	0 ms	200 ms	0	0
		DWH Customers Dimension	process	0 ms	416 ms	500	0
		Get No of City ZIP Codes	process	0 ms	349 ms	0	0
		US Customer Data	process	0 ms	66 ms	0	500
	Total			4,133 ms	1,081 ms	500	1,150
2005-09-12 08:23:59.765	Demo Transfer US Customers		execute project	4,233 ms	0 ms	0	0
		California ZIPs	process	0 ms	83 ms	0	650
		Compute City Size & Customer Fullname	process	0 ms	199 ms	0	0
		DWH Customers Dimension	process	0 ms	399 ms	500	0
		Get No of City ZIP Codes	process	0 ms	365 ms	0	0
		US Customer Data	process	0 ms	66 ms	0	500
				4,233 ms	1,112 ms	500	1,150

Graph

Figure 5-3: Example of graph report capturing project execution time and records moved



CHAPTER 6

Components

Overview

The Sybase ETL SBE components are used to create projects and jobs. (See Chapter 4, “Projects and Jobs.”) They are located in the Component Store. Project components are divided into the following:

- Source components
- Transformation components
- Lookup components
- Staging components
- Destination components
- Job components

Source components deliver data for a transformation stream. A project has to start with one or multiple Source components. This component type has no IN-port and one or more OUT-ports.

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

Destination components (also called data sinks) write data to specific targets. This component type has one IN-port and no OUT-port.

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

Setting up a component

Each component is dedicated to a specific task and therefore incorporates task-specific features. However, all components that connect to database objects follow the same procedure to set up the database connection parameters. This chapter summarizes common characteristics that are available for most components. When in doubt, refer to the detailed description of the specific component later in this chapter.

Setting required properties

Before a component is ready to be used within a project, you must set the required properties. In some cases, adding a component to the Design area opens a window where you can set the values of required parameters. All required property labels are bold.

Data blocks and visualization

The data stream that is passed from component to component during a step-by-step simulation is adaptable and highly transparent. The number of records that are being processed within a step can be configured. You can preview data at any stage by previewing the content of a Link or by looking into the component itself.

Ports and links

A component has IN-ports to receive data and OUT-ports to pass the data after it has processed it. When stepping a component, the data forwarded through the OUT-port is delivered to the IN-port of the subsequent component.

Ports pass data through connecting links. Every link has the capability to map the fields of the ports it connects to. Ports also indicate the status of a component (green for “ready-to-use”, red for “missing or invalid mandatory property settings”).

Repetitive stepping

During a simulation the component can be executed (stepped) separately from its adjacent component. Repetitive steps will not increase the number of records delivered to the downstream component.

Entering database connection parameters

When you add a component with database connection properties to the Design section, the Database Configuration dialog box is opened. The dialog box gives you a central place to enter the most common parameters to set up a connection. More parameters are available in the Property section once the Database Configuration dialog is completed. This section explains all properties of a data provider component.

❖ **To enter parameters**

1 Click the Interface arrow to select one of the following types of database interfaces:

- Sybase

  Note Sybase Open Client must be installed on the same machine as Sybase ETL SBE Development desktop and the ETL Server must be defined in the *sql.ini* file. If the component is to be sent to an ETL Server, then the ETL Server must also have access to Open Client libraries.

- SQLite Persistent

  Note Sybase ETL SBE ships with a built-in, general purpose, relational database that is based on SQLite (www.sqlite.org).

- IBM DB/2
- ODBC

  Note The ODBC driver must be installed on the same machine as Sybase ETL SBE Development desktop and a system data source name (DSN) must be defined for the target. If the component is to be sent to an ETL Server, then the ETL Server must also have access to the proper ODBC drivers and DSN.

- Oracle

2 Click the Host Name list and select one of the available host names.

If you are interfacing to a SQLite Persistent interface, you can enter a database file name.

3 Enter a valid database user and password combination for your connection if required.

The value provided for the password is encrypted and saved permanently to avoid repetitive entry.

4 Click Query to open the Query dialog box. (Refer to Chapter 5, "Advanced Concepts and Tools" for a detailed description of the Query Designer/Content Explorer.)

5 In the Database field, enter a database name, if required.

6 In the Schema field, enter a database schema name, if required.

7 Activate the standardize data format option.

When you activate the Standardize Data Format in a Data Provider component, incoming DATE and NUMBER information is automatically converted into a standardized format (Date: CY-M-D H:N:S.s; Example: 2005-12-01 16:40:59.123; Numbers with a '.' as decimal separator). This allows to automatically move and convert this format sensitive information between systems using different DATE and NUMBER information.

When activated in a Data Sink component, the component expects to receive all data in attributes of the datatypes DATE or NUMBER in the standardized data formats. The data sink component then automatically converts from the standardized data format to the native format of the connected database system.

8 Click Edit to open the Database Options dialog box and select the database options.

DB Option	Default	Description
Show error location	1	1 = yes 0 = no Database errors will include the position of the record within the result set, when Show error location = 1.
Always use logon credentials	0	Always use logon credentials.
Extended connection option		Extended connection option.
Connect timeout	0	Stops trying to connect after Connect timeout seconds. If set to 0, the connect will not time out
Disconnect timeout	10	Enforces disconnect from the database, if there was no reply from database *n* seconds after trying to disconnect.
Treat numeric values as character	0	1 = Treats numeric values as characters.
Always Unicode	0	Unicode will be forced (DB2).
Isolation Level	DEFAULT	Database specific.

DB Option	Default	Description
Lock result set data	0	Query tables will be locked. This is used to ensure that no data is written to the selected record set while the process in working on it. The selected record set is released when the last record from that set was fetched. This can be useful when dealing with data integrity concerns.
LOB truncate size	1024	LOB will be truncated when exceeding LOB truncate size.
Numeric Support	1	Support numeric values.
Execution timeout	0	Component will stop execution after a time interval in seconds (0 = no timeout.)
Unicode support	0	Support Unicode operation.
Write rejected records to file		File path for Reject Log. This and the following options are used to log records that have been rejected by the database on loading **Note** All other records of a data block will be written to the database
Truncate reject log	1	Log will be truncated on database connect. A value of 0 will append data to an existing log file.
Write error code to reject log	1	The database error code for each record will be written to the log.
Write error text to reject log	1	The database error text for each record will be written to the log.
Write header to reject log	0	If set to 1, a column header will be written to the log.
Reject log column delimiter	Tab	The columns will be delimited by this character or string.

9 Set the Read Block Size.

The Read Block Size option defines the number of records retrieved by the component within in a single step.

Use a fairly small number during simulations to accelerate the simulation step. Increase the number before you execute the project. Executing a project with a small number might have a negative effect on the overall performance of the system.

10 Set the Write Block Size.

The Write Block Size option defines the number of records to be written to the database in a single write operation. A component with a large Write Block Size will be waiting to receive as many records as defined before the data records are actually written to the database. To enforce writing after a simulation step, the number for Write Block Size should be equal or smaller than the Read Block Size of the previous component with a Read Block Size property.

11 Set up pre-processing SQL.

One or more SQL statements can be executed during the initialization of a component. Initialization is the first step each components passes when starting a simulation or when executing a project. When the project is initialized, all components of the project are initialized subsequently.

Note When using multiple statements, you can separate the statements by using a semicolon.

12 Set up post-processing SQL.

One or more SQL statements can be executed when all components finished processing the project.

Note When using multiple statements, you can separate the statements by using a semicolon.

Working with the SQLite Persistent interface

Sybase 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.

Connecting to a SQLite database

A SQLite database is represented as a single file with the extension *.db*. The database file can contain any number of tables.

❖ **To connect to or to create a SQLite database**

1 Select SQLite Persistent from the Interface drop-down list.

2 Enter a new or existing file name for the SQLite database in the Host Name property. A new name will automatically create a new SQLite database file with an extension *.db*. Do not enter the extension *.db* when entering the name of an existing SQLite database.

For 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

- Right click on a Staging component and select Add Staging table from input or Add Staging table from port.

  – Or –

- Right click on 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.

SQLite supported SQL commands can be used in the pre-processing or post-processing SQL properties of components connected to databases.

You can use the Content Explorer from the Tools menu to manipulate or browse objects of the SQLite database connected to components in your project. Also, use client applications available from *sqlite.org* to connect to SQLite database files.

Note Before using external client applications to connect to your SQLite database files, become familiar with the locking strategy of SQLite.

Providing descriptions for components

You can assign a name and a description to a component. The Name appears at the top of the component. The Description and the Name are displayed in a tooltip.

❖ **To provide a description for the component**

1 Right-click the component to open the pop-up menu.

2 Select Description.

3 Enter a name.

4 Enter a description. You can use HTML formatting tags to format the description.

Adding component variables to a component

In the Property section, you can add component variables to your component. Component variables can be seen as parameters to the component. Those variables can be accessed in expressions or procedures of the component for further processing. The scope of the variables is local to the component.

❖ **To add a component variable**

1 Right-click the Property section to open the pop-up menu.

2 Select Add.

3 Enter the name of the variable. Inside the component, this variable is referenced using the notation REF.<*name of variable*>.

4 Enter values for a prompt and a description.

❖ **To edit a component variable**

1 Right-click the component variable in the Property section to open the pop-up menu.

2 Select Edit.

3 Apply any modifications to your current settings.

❖ **To remove a component variable**

1 Right-click the Component variable in the Property section.

2 Select Remove.

Evaluating SBN expressions

Select the Evaluate command if you want to allow SBN expressions that are evaluated prior to using the property value. Some property items are pre-set to Eval.

❖ **To select or clear the Evaluate property**

1 Click the line of the property item you with to disable the evaluation for.

2 Right-click to open the pop-up menu.

3 Click Evaluate to toggle the current setting.

Note If the Evaluate option is set on Password properties, the value will be displayed as plain text, not in password font.

Encrypting properties

Property values are stored in the repository in XML format. However, most entries in the Sybase ETL SBE repository are not encrypted but are represented in a readable character set.

❖ **To save property values in an encrypted format**

1 Click the line of the property.

2 Right-click to open the pop-up menu.

3 Click Encrypt to toggle the current setting.

Note If the Evaluate option is set on Password properties, the value will be displayed as plain text, not in password font.

Modifying components

To initialize a component, right-click it and select Initialize or select Initialize and Step from the pop-up menu.

If you modify one of the existing property settings of a component during simulation re-initialize the component prior to applying the next step.

Data blocks and visualization

When working in simulation mode, the entire result set of the data source (as defined by a query in the source or staging component) can be divided 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 in the Property section of the component. Choose a small number for the Read Block Size parameter to enhance the performance while stepping through the project.

❖ **To preview the result of a transformation**

1. Step through the project including the component you want to preview.
2. Right-click the component, port, or the connecting Link.
3. Select the Preview command. If a component has multiple ports, select the port from the drop-down list.

Stepping a component multiple times

A component can be stepped multiple times during a simulation to allow you to preview a component's behavior with different property settings.

❖ **To step through a component multiple times**

1. Select the component.
2. Modify the transformation rules or property settings.
3. Initialize the component.
4. Step the component.
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 pop-up menu from the desktop.

Managing port structures

When adding a new component to a project, Sybase ETL SBE assigns a port structure to the newly added component based on the connecting link. Adding a component to a project with no unconnected ports leaves the newly added component without a port structure.

Additional attributes can be added to a port structure.

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

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

Modifying a port structure

The structure of a port can be modified in the Structure Viewer window.

❖ **To modify a port structure**

1 Right-click the port.

2 Select Edit Structure to open the Structure Viewer.

Note The port structure of Source Components and Destination Components cannot be changed.

❖ **To add an attribute to the port**

1 Click Add.

2 Enter the attribute settings.

3 Click Save to confirm.

❖ **To delete an attribute from the port**

1 Click the line containing the attribute.

2 Click Delete.

3 Click Save to confirm.

❖ **To modify an attribute**

1 Click on the line containing the attribute.

2 Modify the attribute settings.

3 Click Save to confirm.

Copying port structures from other ports

A port structure can be assigned to a port based on any other available port in the current project:

- Click the port you want to assign a new structure to.
- Right-click and select Assign 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.

❖ **To select 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.

Viewing and mapping to ports

A link is the connecting line between two component ports. A link can be used to:

- Change the mapping between the port structure of the adjacent ports
- Preview the data (before mapping occurs)

❖ **To view the mapping of a link**

1 Right-click the link. The color of the link turns green.

2 Select the Mapping command to open the Mapping Definition window.

In Display structure mode, all attributes of the connected port and their current mappings are shown. The Display structure and values mode displays the value of the current record.

❖ **To change a mapping**

1 Click on a connecting line.

2 Remap it to a new port attribute by dragging the line end to a new connection point. (See “Viewing current mappings” on page 28.)

Source components

Source components deliver data for a transformation stream. A project has to start with one or multiple Source components. This component type has 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.
XML via XSLT Data Provider	Applies an XSLT transformation on an XML document and outputs the results as a single column record.

DB Data Provider Full Load

DB Data Provider Full Load extracts data from any data source accessible by an ODBC connection or a native driver (DB2, Oracle, Sybase). This component issues a query against the data source to retrieve the data. The structure of the (single) output port mirrors the structure of the query result set.

DB Data Provider Full Load is one of the source components that initiates the transformation process. You can use this component whenever the source database has a relational structure and is accessible through a supported driver.

Adding DB Data Provider Full Load to a project

Adding DB Data Provider Full Load to a project opens a configuration window and properties section. Use the configuration window and properties section to define connection parameters and query options.

❖ **To enter required properties**

1 Enter the connection parameters as described in "Entering database connection parameters" on page 68.

2 In the properties section, click Query to open the Enter Property window.

Adding a Select statement to the Query field

You can create a query by entering a SELECT statement into the Query field. Any valid SELECT statement based on the underlying database system and schema is allowed. No delimiting character to mark the end of the statement is required, for example:

```
SELECT * FROM CUSTOMERS
SELECT A.CU_NO, A.CU_NAME, A.CU_CITY, B.SA_ORDERDATE,
B.SA_TOTAL
FROM CUSTOMERS A, SALES B
WHERE A.CU_NO = B.CU_ID
ORDER BY A.CU_NO
```

For database schema lookups, click Database Lookup.

Note The SELECT statement must be compliant with the SQL language supported by the connected database system. Be especially aware of differences regarding quote characters, date functions, and expressions. It is good practice to use the SQL 92 ANSI standard for queries compatible between different database systems.

Using Query Designer

Click Query Designer to open the Query Designer. To learn more about this tool, see "Query designer" on page 52.

Validating a query

Click Execute Query to run the query.

Modifying a connection

After modifying connection parameters, re-logon to the database to validate the settings. To re-logon, right-click the component in the project and select Logon.

Optional properties

- Read Block Size — defines the number of records retrieved by the component within a single simulation step.
- Pre-processing SQL — defines one or more SQL statements to be executed during initialization of the component.
- Post-processing SQL — defines one or more SQL statements to be executed after all components finished execution.

See "Entering database connection parameters" on page 68 for more information about the optional properties and for more information about the following properties:

- Database
- Schema
- Database options

Impact on simulation sequence

Read Block Size value impacts the number of records loaded in a single simulation step.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Transfer German Customers
- Demo Transfer German Products
- Demo Transfer German Sales
- Demo Transfer US Customers
- Demo Transfer US Products

DB Data Provider Index Load

DB Data Provide Index Load performs incremental data loads. The incremental load is controlled by an Ascending Index attribute that contains the ascending values. DB Data Provide Index Load ignores any data records already extracted by previous executions of the project.

Use this component to track source changes on a regular basis. You can significantly reduce load time if only the delta of the day has to be transferred, instead of loading very large tables or views on a daily basis.

Adding DB Data Provider Index Load to a project

Adding DB Data Provider Index Load to a project opens a configuration window. Use the configuration window and properties section to define connection parameters, queries, and Ascending Index attributes.

❖ **To enter required properties**

1 Enter the Connection Parameters as described in "Entering database connection parameters" on page 68.

2 Select the Ascending Index attribute from the list of database objects.

3 Select an attribute with values that increase whenever data is changed or added to the source, such as an auto incremental ID or a modification date. It is not required to have this attribute indexed on the database schema level; however, for performance reasons Sybase recommends that you create an index.

4 Enter a Query for the incremental load.

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.

Resetting the Ascending Index Value

You cannot directly manipulate the persistent value of the Load Index in the Repository, but you can reset the value to the one stored in the Load Index Value property.

❖ **To reset the value of the persistent Load Index value**

1 Select the project in the Navigator section.

2 Right-click and select Reset Execution Properties.

Optional properties

- Load Index Value - The maximum value of the Ascending Index attribute is automatically used and stored when executing a Sybase ETL Small Business Edition project, 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 — The Read Block Size defines the number of records retrieved by the component within in a single simulation step.
- Pre-processing SQL — defines one or more SQL statements to be executed during initialization of the component.
- Post-processing SQL — One or more SQL statements to be executed after all components finished execution.

See "Entering database connection parameters" on page 68 for more information about the optional properties and for more information about the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on simulation sequence

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

See also

For more information, see *Demo Transfer US Sales on an incremental basis* in DemoRepository and Help Flash movie.

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 fields or fields delimited by unique field delimiters.

Use this component to extract data from text files with fixed length or delimited fields.

Adding Text Data Provider to a project

When you add Text Data Provider to a project, Sybase ETL opens a component window and prompts you for the source file. Identify the source file, then use the component window to identify properties about the source data.

❖ **To use the Text Data Provider**

1. Add the component to the project. The component window and the Open File dialog box open.
2. Select a file or close the window to enter a name in the Text Source field. After select the file, you are prompted to select file description properties.

   While reading the file, Sybase ETL Small Business Edition tries to make best assumptions about the structure of the file, but any value can be modified.
3. The following message appears `"Do you want to generate the column definition?"`. Click Yes if you want to create columns based on the current settings. Click No to create a single column.

Properties

Character Encoding

- Select a character set from the list.
- Press Enter to confirm.

Support Unicode

Activate this option when the text file contains unicode data.

Type

- Specify the structure type of the input file.

- Press Enter to confirm.

Delimited

The fields in the file are separated by a specific character or string.

Fixed (fixed Line)

The fields and lines in the file are of fixed length. There is no line delimiter.

Fixed (variable line)

The fields in the file are of fixed length. The lines are separated by a specific character or string.

Line delimiter

- Either select from the list or enter a value.
- Press Enter to confirm.

Line length

- Enter a number that matches the line length of a Fixed (Fixed Line) file.
- Press Enter to confirm.

Column delimiter

- Either select from the list or enter a value.
- Press Enter to confirm.

Read column names from row

- Enter the number of the line containing the column headings.
- Press Enter to confirm.

Skip first rows

- If any rows are to be skipped on reading (for instance lines containing headings), provide the number of lines to be skipped.
- Press Enter to confirm.

Quote characters

- If field values in the source text file are quoted, provide the respective character.
- Press Enter to confirm.

Null byte replacement

- Enter the character to replace a Null Byte (0x00 ASCII).
- Press Enter to confirm.

Modifying column names

- Right-click the current column name in the Output Port Content area and select Edit.
- Modify the Column Name. For a delimited file, no position parameters will be displayed.

Removing columns

To remove columns, right-click the column name in the Output Port Content area and select Remove.

Working with fixed-length file type

When a file has been qualified as fixed-length, the Add Column option in the Output Port Content area is activated. For each column to be added, click Add Column and edit Column Name, From Position and To Position. Or, you can select the range of the column to be specified in the upper File Content section.

Right-click and select Assign Position to Column. Then, enter the Column Name. The position parameters are preset according to the position of the selected area in the line.

Impact on Simulation sequence

There is no impact on Simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

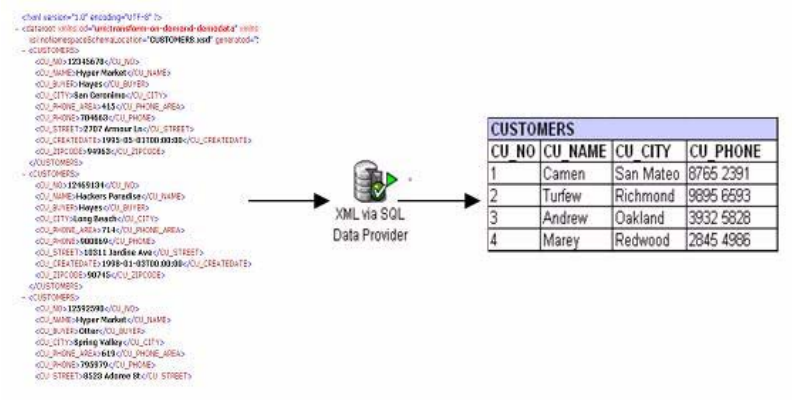
- Demo Transfer German Customers

- Demo Transfer German Sales
- Demo Transfer US Customers
- Demo Data Transposer

XML via SQL Data Provider

XML via SQL Data Provider loads hierarchical XML data into a relational schema. You can use SQL to query the schema like a relational database.

Figure 6-1: XML via SQL Data Provider



Use the SQL-based XML Port Manager component to define one or more output data streams. You can provide either an XML Schema file or DTD to validate the XML source.

You do not need to understand XML in detail to use this component. It is useful for data-centric XML documents (such as, sales order, stock quotes, scientific data) that are characterized by a regular hierarchical structure. Use this component to represent the data from the XML document in a relational table structure for further data transformation.

Adding XML via SQL Data Provider to a project

When you add XML via SQL Data Provider to a project, Sybase ETL opens a properties section and prompts you for the source file. Identify the source file, then assign the data output properties.

❖ **To enter required properties**

1. Select the XML Source file. This can be an HTTP, FTP, URL, or a file name.
2. Click the XML button of the Data Output property to open the XML Port Manager.
3. Specify SELECT statements based on the table structure created from the XML document for each output port (see description of the XML Port Manager below). There is one output port per default, but you can add ports. To create the SELECT statement, you can use the Query Designer.

Optional properties

Document Schema

Enter the URL of an external schema or DTD for the XML document.

Namespace Schema

Enter the URL of an external schema for the namespace.

Validate Schema

Switch on or off schema/DTD validation.

XML Options

- Process namespace — Set this value to 0 if namespace specification shall not be considered during parsing. The default value is 1.
- Full schema check — Full schema constraint checking includes those checking that may be time-consuming or memory intensive. Currently, 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

Select the file containing the database schema setup (create tables) script. Use this option to enforce a fixed data model.

DB Schema Options

Customize the settings for tables and attributes being generated from the XML structure, such as the prefixes for table and attribute names.

DB Options

These are advanced options to configure the underlying internal database.

Read Block Size

This option defines how many records are read from the query result set in one step. It applies only to components with a single output port in the simulation. With a value of 0 (default) all records are delivered in one output block.

Setting up the XML via SQL component

This section guides you through the three steps of the setup process for the component using a simple example.

To follow the example, use the *PRODUCTS.xml* file as the XML source. It is located in the Demodata subdirectory of the Sybase ETL Small Business Edition installation directory.

XML source

The following XML document is a simple product structure. Each product is described with an ID (PR_ID), a name (PR_NAME), a product group (PR_GROUP1), and a 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>
```

XML Port Manager

Click the XML button of the Data Output property to open the XML Port Manager.

In the upper area of the window, you can view the source XML document. The left section of the lower window contains two tabs:

- The Data Model tab displays the generated relational schema.
- The Reference tab displays the available component variables.

The right section of the lower window is the port area in which one or more ports can be defined. Each port is described by a SELECT statement based on the XML Data Model tables.

The Data Model

There is one table for the root element (TAB_dataroot) followed by one or more tables for elements on level 1. In the example, only one element on this level exists (TAB_PRODUCTS). On 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, each level will create another set of table.

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 via attribute ATT_FK_dataroot. The tables on level 2 are linked to table PRODUCTS via attribute ATT_FK_PRODUCTS. To create the view containing the PRODUCTS records, the tables on 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
```

Creating a SELECT statement

You can enter a SELECT statement for the port straight into the port field, or you can open the Query Designer by clicking Query Designer.

Note To control the automatic join generation (enable/disable), go the File | Preferences menu.

Adding and removing ports

Right-click on the port section and select Add port or Remove port.

An Info Port can be added to forward the XML document to the next component. This port is visible after exiting the XML Port Manager.

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo XML CNN RSS News
- Demo XML via SQL Data Provider

XML via XSLT Data Provider

XML via XSLT Data Provider applies an XSLT transformation on an XML document and outputs the results as a single column record. XML via XSLT Data Provider requires an XSLT stylesheet to process an XML document. You could, for example, apply an XSLT stylesheet to an XML document to create HTML output.

Adding XML via XSLT Data Provider to a project

Adding XML via XSLT Data Provider to a project opens a file selection window you use to identify the XML source.

❖ **To use the XSLT Data Provider**

1 Select or enter the URL to the XML source file.

2 Click on the Stylesheet edit button to open the XSLT editor.

3 Enter the XSLT stylesheet script into the XSLT Content area.

Impact on simulation sequence.

There is no impact on the simulation sequence.

See also

For more information, see *Demo XML via XSLT Data Provider* in DemoRepository.

Transformation components

Transformation components, Lookup components, and 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
Data Calculator JavaScript	Performs transformations to every record passed to this component between the IN- and OUT-port.
Data Splitter JavaScript	Splits an incoming data stream into two or more outgoing data streams.
Character Mapper	Replaces characters and strings in an input record. Character Mapper applies replacement mapping to all selected attributes. You can save character mapping definitions into files.
Data Mapper	Data Mapper merges two data input streams with the same or different record structure into a single data output stream.
Data Transposer	Creates multiple output records from a single record. You can use this component to copy a single input record into multiple output records and assign a different mapping to each output record.

Data Calculator JavaScript

Data Calculator JavaScript performs transformations to every record passed to this component. You can freely map and transform attributes between the IN- and OUT-port. When you enter your transformation rules, Data Calculator allows you to verify the incoming content, applied transformations, and outgoing content based on the proposed rule set.

Use Data Calculator to transform the content of one or more port attributes or add rules to for new attributes. To remap attributes, use the mapping feature of a Link without having to use a Data Calculator.

Adding Data Calculator to a project

Adding Data Calculator to a project opens a component window. You must create an initial mapping between the IN- and the OUT-Port. The two most common mapping methods are available through the Create mapping by order and Create mapping by name buttons.

❖ **To create a mapping**

1. Select the Graph tab of the component window.
2. Map the IN-Port and Out-Port structures by either using pre-defined mapping sequences or by connecting the IN-Port and OUT-Port attributes individually.

After you select a mapping sequence, the port color changes to green. The component is now ready to be used and will forward records from the IN-Port to the OUT-Port.

Displaying transformation results

One of the essential features of the Data Calculator is its capability to display the result of transformation rules immediately. This powerful simulation capability allows you to verify the incoming data, the transformation rules, and their effect on the data output. By changing the Current Input Record you can even enter your own test data for verifying your transformation rule.

The Data Calculator window consists of two tabbed section: Tabular and Graph. When the Data Calculator window opens, it displays the tabular view of the current simulation.

Tabular tab

There are four major areas in the Tabular tab:

- Current Input Record
- Transformation Rules
- Current Output Record
- Input/Output Port Content

Current input record

This area displays the content of the current record. You can freely change the values of each attribute and verify the result in the Transformation Rules and Current Output Record area. Manual modifications made in this area only affect the data of the Input/Output Port buffer. This offers a convenient way to create test cases for transformation rules.

❖ **To simulate a single attribute**

1. Change the value of any attribute in the Current Input Record.

2 Press Enter.

The result of transformation based on the new field value is displayed in the area to the right.

Transformation rules and current output record

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 would look like the following example:

3	IN.PR_PRICE		OUT.PR_PRICE	low end
4	IN.PR_GROUP1		OUT.PR_GROUP1	CD Rom
5	uUpper(IN.PR_NAME)		OUT.PR_NAME	24 CD ROM DRIVE
6	IN.PR_ID		OUT.PR_ID	435672
7	if (IN.PR_PRICE < 250) OUT.P			
8	'valid'		OUT.PR_DESC	valid

You can use the Procedure Editor (see “Using the JavaScript Procedure Editor and Debugger” on page 54) to create more complex procedural transformations.

The attributes displayed after opening the Data Calculator window in the Tabular view are related to attribute structure of the IN-port and the OUT-Port.

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

Note The view in the Graph Tab mirrors the actual Port Structures. You cannot add or delete attributes there.

❖ **To manually add transformation rules**

1 Right-click anywhere in the Transformation Rule and Current Output Port column.

2 Select the Insert command to add a line.

3 Click the added line.

4 Right-click and select OUT or TMP.

5 Select one of the attributes from the menu.

Other options

- The added attribute can now be used for further assignments or calculations.
- To delete transformation rules, right-click the desired line in the Transformation Rule or Current Output Port column and select Remove.
- To change the order of the transformation rules, right-click the desired line in the Transformation Rule or Current Output Port column and select UP or DOWN.

Sequence of processing transformation rules

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

In Line 3 you find a direct mapping between IN.PRICE and OUT.PRICE. Later in line 7, there is a procedure that re-calculates the price and is thus overwriting the assignment of the previous transformation rule.

Input Port Content area

Shows the current set of records available at the input port.

Output Port Content area

Shows the current set of records available at the output port.

Graph tab

The Graph tab shows the current mapping between the IN- and OUT-port attributes.

You can freely change the mapping by re-connecting the lines to the connection points.

Note After you apply a transformation rule to an IN.attribute, the mapping line between the IN.attribute and the OUT.attribute will no longer be displayed.

Simulating Data Calculator

Data Calculator is designed to visualize the changes applied to the data as it moves through the transformation rules. This is particularly useful to answer the question on how a change of a transformation rule would affect the outgoing data. Depending on status of the Auto-Synchronization button, a transformation rule will be immediately applied to the entire set of IN-Port records after the rule had been entered.

Turn on and off immediate synchronization

Selecting Auto Synchronisation immediately applies all changes to the transformation rules to the current set of records at the IN-Port.

To switch the current setting, click Auto-Synchronization. If Auto Synchronization is turned off you can manually trigger the processing of the IN-Port data by select the Step option.

Manually Apply all transformation rules

To manually apply all transformation rules to all current records at the IN-Port, click the component Step option.

Fetch another set of records

To step through the next incoming data buffer set, click Fetch the Next Set of Data.

Select a specific record

To step through the current set of records at the IN-Port record-by-record, click the appropriate option in the record control. Or, you can select a record from the Input Port Content list.

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

Search for keywords in the transformation rules

Click the Search the Content of Transformation Rules to search for a specific keyword in the current transformation rules.

Highlight NULL-Values and empty values

For easy identification of NULL-values and empty values, click Highlight NULL-Values and Empty Values.

Using lookups in Data Calculator

Data Calculator can perform lookups on the attribute level. The lookup data has to be provided at special lookup ports.

Adding Lookup Ports

To add a lookup port to a Data Calculator component, connect the output port of the data providing component with the Data Calculator component. Or, you can select Add Input Port from the Data Calculator component menu and connect the output 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 the return values.

If you need to lookup compound keys you have to 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

You can use the Lookup Options property to configure the lookup. The Enter Property window will display a list of all lookup ports and its current option values.

The properties are:

- Lookup Name — The 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 (this option is described more detail in “DB Lookup” on page 110).
- 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 the appropriate options.

Building Lookup rules

To set up Lookup rules open the Tabular Tab of the Data Calculator window. If Lookup ports are available an additional Lookup column is displayed.

You must 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 columns you have to specify which value to return, if the key has been found. This is done by selecting the associated port attribute from the pop-up menu on the Lookup column (for example LOOKUP1>>LOOKUP1.PR_NAME).

Note Although the return values are selected and displayed by name, the lookup internally uses the column number. This means you will need to review your lookup rules whenever the lookup port structure is modified, especially after adding or removing attributes.

Note The key column is always returned as Unicode.

- 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 will return Null if the given key value is not found. To return a different default value you can enter an expression in the Lookup Options window (for example, “key not found”).

Impact on a simulation sequence

Without Lookups, there is no impact on a simulation sequence. With Lookups, all data is read into the Lookup ports before the data at the Main port is processed.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Data Calculator
- Demo Transfer German Customers (with Lookup)
- Demo Transfer German Products
- Demo Transfer German Sales (with Lookups)
- Demo Transfer US Customers (with Lookup)
- Demo Transfer US Products

Data Splitter JavaScript

Data Splitter JavaScript splits an incoming data stream into multiple outgoing data streams. Although you can add additional ports, Data Splitter is configured with two OUT-Ports by default. The distribution of records is controlled by port conditions. You can define overlapping port conditions that are not mutually exclusive.

Note The number of records forwarded by the Data Splitter to the OUT-Ports can be greater than the number of incoming records. If a single record matches more than one port condition, it will be available on all of these ports.

Use Data Splitter JavaScript to split up (or copy) a data stream of records according to user defined conditions.

Adding Data Splitter to a project

Adding Data Splitter to a project opens a property window. Use the property window to define rules and port conditions.

In the properties section, click the button in the Value column of the Rule property to open the Data Splitter component window. Data Splitter is configured with two default ports. Port conditions for the OUT1 and OUT2 port are pre-set to 1. Because this condition is always true (regardless of the current values if the IN port), Data Splitter will copy all incoming records to both output ports.

The current set of input data is shown in the upper part of the component window. Selecting a record in the upper window causes the lower portion of the window to signal if the port conditions are met (green) or not (red).

Customizing port conditions

You can assign a condition to each port. A condition consists of one ore more expressions. Multiple expressions are concatenated by operators. The result of a condition is either TRUE (1) or FALSE (0).

A condition consists of functions and logical operators.

❖ **To modify a port condition**

1 Click the port.

2 Click the Edit calculation button to open the Condition window.

3 Enter the condition in the right section of the window. The left area of the window contains two tabs: Variables and Functions. In the Variables tab all variables are listed that may be used in the condition. The Functions tab lists all functions and operators.

To add a port, click the Add button. To remove a port, select the port and click the Remove port button.

Special port conditions

The following are special port conditions:

- 1 — TRUE, all records will be 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.

Current record conditions

When you select a record of the input buffer in the upper part of the Data Splitter window, the ports in the lower half of the window will be updated. If the selected record matches a condition, the port light will turn into green. If the selected report does not match the port, the port will turn into red.

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

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

Character Mapper

Character Mapper replaces characters and strings in an input record. Character Mapper applies replacement mapping to all selected attributes. You can save character mapping definitions into files.

Use this component for replacement of characters or strings (for example, German Umlaut ä to ae or Unicode characters).

Adding Character Mapper to a project

Adding Character Mapper to a project opens a properties window that allows you define the character mappings. Use the options in the properties section to define the character mappings.

❖ **To set up required properties**

1. Add the component to the project and connect the ports of the component to adjacent components.
2. Open the Mapping window by clicking Mapping in the properties section.
3. Enter at least one mapping.

❖ **To add a character mapping**

1. Click Add Line in the toolbar.
2. Enter a character or string in the From columns.
3. Enter a character or string in the To column. If you do not enter a value, the From value will be effectively removed without any replacement.

The Character Mapper component is designed to visualize the result of a character mapping immediately after the mapping has been entered. To switch off this kind of automatic synchronization, click the Auto synchronization button.

Other options

- To delete a character mapping, select the line and click the Remove Line button.
- To save the current mapping to a file, click the Save File button.
- To open a mapping from a file, click the Open File button.

Optional properties

In the properties section, you can select the attributes to use the character mapping previously defined. By default the mapping rules will be applied to all attributes of the incoming data.

To customize the default setting, click Exclude and then select the attributes that you want to exclude from the mapping.

Example values for a character mapping are:

From	To	Description
Ä	ae	Umlaut character mapping.
customer	kunde	String replacing.

From	To	Description
<13><10>		(ASCII decimal) Delete CR LF at the end of the line.
<xD><xA>		(ASCII hexadecimal) Delete CR LF at the end of the line.
<u8356>	<u8364>	(Unicode Decimal) Replace the Lira currency symbol with Euro currency symbol (€).
<0x20A4>	<ux20AC>	(Unicode Hexadecimal) Replace the Lira currency symbol with Euro currency symbol (€).

Impact on simulation sequence

There is no impact on the simulation sequence.

See also

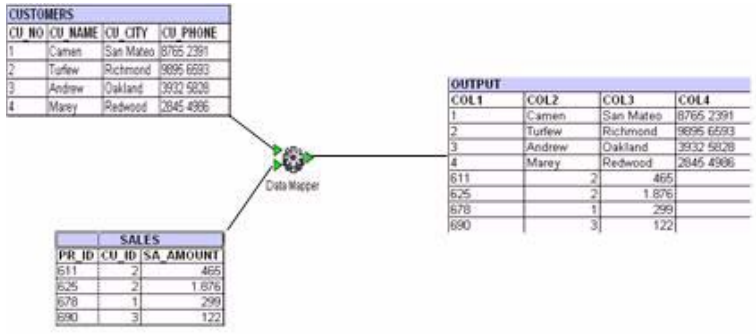
For more information, see *Demo Character Mapper* in DemoRepository.

Data Mapper

Data Mapper merges two data input streams with the same or different record structure into a single data output stream.

Use this component to merge two data streams into a single data stream. This allows you to concatenate data from different sources, but related to each other, into a single data stream.

Figure 6-2: Data Mapper



Adding Data Mapper to a project

Adding Data Mapper to a project opens a component window. Data Mapper includes two IN-Ports and one OUT-Port. You can map each input data stream to the OUT-Port structure.

❖ Using the Data Mapper

1 Add the component to the project and connect the ports of the component to adjacent components.

2 Click the Rule button in the Property section to open the Data Mapper window.

3 Enter a mapping between IN1-Port and OUT-Port. Use the Enforced Output Value to assign computed expressions to the OUT-Port attribute.

4 Enter a mapping between IN2-Port and OUT-Port. Use the Enforced Output Value to assign computed expressions to the OUT-Port attribute.

5 Create the mappings for IN1-Port and IN2-Port.

The Data Mapper window contains of two tabs, IN1 and IN2 (located at the upper right corner of the mapping area). IN1 tab allows to map the upper IN-Port to the OUT-Port. The IN2 tab contains the mapping between lower IN-Port and the OUT.-Port.

The IN1 and IN2 tabs are located in the upper right corner of the component window.

You can create the mapping either automatically by using the Create mapping by order or Create mapping by name button. Alternatively you can draw the mapping lines manually.

Enforced output value

Instead of mapping one of the input record attributes to the OUT-Port structure, you can assign expressions to one or more OUT-Port attributes. An expression can be either a literal (for example, “Quarter”) or a calculation. An expression can also contain a SBN expression.

Example for Square Bracket Notation (SBN):

```
[uDate("now")]
```

Impact on the Simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Data Mapper
- Demo Transfer All Customers

Data Transposer

Data Transposer creates multiple records from a single record. You can use this component to copy a single input record into multiple output records and assign a different mapping to each output record.

Use this component to normalize record structures.

Adding Data Transposer to a project

Adding Data Transposer to a project opens a component window. The Data Transposer component has one IN-Port and one OUT-Port. All generated output records are forwarded to the same OUT-Port. This feature allows to normalize data structures typical to mainframe files or spreadsheets.

❖ **To use the Data Transposer**

1 Create a default mapping between the IN-Port structure and the OUT-Port structure. This default mapping is shared for subsequent mapping tabs. A default mapping is optional.

2 Create new tab. Creating a new tab will create an additional output record.

3 For the tab created in the previous step, map the IN-Port structure to the OUT-Port structure. Use the Enforced Output Value column to assign expressions to the OUT-Port attribute.

   A new output record is created by pushing the Insert Button located on the toolbar of the Transposer window. This will generate a new Tab in the Tab-section of the window. If a default mapping was created, each generated output record will inherit this mapping. The Tab-section consists of the Tab Default and n Tabs, where n is the number of copies that will be generated for each single input record.

   After connecting the component to the adjacent components:

4 Check the structure of the IN-Port and OUT-Port.

5 Open the component window. The current IN-Port and OUT-Port structure is displayed in the Default mapping tab. If all the OUT-Records share a common mapping, this is the place to assign that mapping.

6 Click the Insert button to create an output record structure. A new mapping area will be displayed. If a default mapping had been assigned, the appropriate attributes will already be connected.

7 Create a mapping between the IN and the OUT structure:

   - Connect two opposite connection points of the Mapping Area.
   - Use an expression in the Enforced Output Value column to assign a calculation, for example: uConcat('Arrival Date: ',uDate('now','localtime'),' ',uTime('now','localtime')).

     Referencing IN-Port variables is not allowed.
   - Go back to step 6 to create more output record structures.

Impact on simulation sequence.

There is no impact on the simulation sequence.

See also

For more information, see *Demo Data Transposer* in the Demo Repository.

Lookup components

In general, a Lookup operation looks up a value corresponding to a key in a Lookup table containing a list of (key, 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
Data Lookup	Data Lookup generates a lookup based on data provided during the transformation process. The component has two input ports: one for the incoming data records and a second port for the lookup data. Use Data Lookup when you cannot retrieve lookup data directly from a database, but can retrieve lookups provided by the transformation process. For example, you may want to use this procedure when data is located in a text file.
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.
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.

Data Lookup

Data Lookup generates a lookup based on data provided during the transformation process. The component has two input ports: one for the incoming data records and a second port for the lookup data.

Use Data Lookup when your lookup data cannot be retrieved directly from a database, but can be provided by the transformation process. For example, you may want to use this procedure when data is located in a text file.

Adding the Data Lookup component to a project

Adding Data Lookup to a project opens a property window. Use the property window to define the Key Attribute, Value Attribute, connection parameters, and query options.

The following procedure assumes that the lookup data is extracted from a text file.

❖ **To use the Data Lookup component**

1 Add the component to the project.

2 Connect the upper IN-Port (Main) with the OUT-Port of the previous component, that will provide the data, containing the key to be looked up, to the Data Lookup component.

3 Add a Text Data Provider component to the project.

4 Connect the lower IN-Port (Lookup) of the Data Lookup component with the OUT-Port of the Text Data Provider component.

5 Specify all data that is required to set up the Text Data Provider Component. Make sure that a proper column definition for the file is provided.

Required properties

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 overwriting a key with its corresponding value.

Optional properties

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

If Use Key Value is activated, the key value will be assigned to the value attribute instead of the default, if the lookup fails.

Lookup Empty/Null Keys

If activated, the lookup will be performed even for empty or NULL key values. Otherwise the selected default method applies.

Lookup Size

Enter the estimated number of lookup records to optimize memory allocation and lookup performance.

Impact on simulation sequence

All data is read into the Lookup port before a lookup is performed on the data at the Main port.

See also

For more information, see *Demo Transfer All Customers* in DemoRepository.

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 tableduring project execution. Changes applied to the underlying database during project execution have no effect on the lookup result.

Use this component, when you can retrieve lookup data directly from a database.

Adding DB Lookup to a project

Add the component to the project and connect the ports to adjacent components. Use the property section to define attributes, connection parameters, and query options.

Required properties

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 returned 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 overwriting a key with its corresponding value.

Enter the Connection Parameters as described in “Entering database connection parameters” on page 68.

Query

To enter the query for the lookup, open the Query window.

The Lookup table (actually, it is a virtual view) is made of a SELECT query with two columns (the key column and the value column). The key column corresponds with the Key Attribute selected. The value column corresponds with the Value Attribute.

To enter the query, enter a SELECT statement that returns the two-column lookup table:

```
SELECT <key attribute>, <value attribute>
FROM <lookup table>
```

You can extend the SELECT command with a WHERE clause or any other valid clause accepted by the underlying database.

Optional properties

Default Value

Specify a Default Value to assign to the value attribute, if a key value was not found in the lookup table.

Use Key Value

If Use Key Value is activated, the key value will be assigned to the value attribute instead of the default, if the lookup does not find a match.

Lookup Empty/Null Keys

If activated, the lookup will also be performed for empty or NULL key values; otherwise, the selected default method applies.

Lookup Size

Enter the estimated number of lookup records to optimize memory allocation and lookup performance.

The entire result set is loaded into cache memory and remains unchanged until the transformation process finishes. Choosing an appropriate value for Lookup Size allows allocating memory in one piece while a low value will cause the program to allocate memory incrementally.

Refer to “Entering database connection parameters” on page 68 for more information on the previous and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Example

Assume that you want to replace the product number used for German products by the product number used in the U.S. The German products are in the table PRODUKTE(PR_NUMMER, PR_NAME, PR_PREIS). The IN-Port of the DB Lookup component contains those three attributes.

The table to perform the lookup of the U.S. product number is 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 Component for this example, choose:

- Key Attribute: IN.PR_NUMMER
- Value Attribute: IN.PR_NUMMER
- Default Value: “INVALID”
- Query: SELECT SOURCE, DESTINATION FROM LOOKUP_PRODUCTS

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

- 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.

Adding DB Lookup Dynamic to a project

Add the component to the project and connect the ports to adjacent components. Use the property section to define attributes, connection parameters, and query values.

Required properties

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 returned 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 overwriting a key with its corresponding value.

Enter the Connection Parameters as described in “Entering database connection parameters” on page 68.

Query

To open the Query window, click Query and enter the query for the lookup.

The Lookup is made of a SELECT query returning a single value that corresponds with the Value Attribute.

The query required in the DB Lookup Dynamic component is slightly different from the one used in the DB Lookup component. The query will return exactly one single value, which is the value found for the corresponding Key Attribute. The value of the Key Attribute is represented in a predefined variable named *LOOKUP*, which is used as a placeholder in the WHERE clause of the query. The notation for this placeholder is an SBN expression.

The basic structure of the query for the DB Lookup Dynamic component is:

```
SELECT <value attribute>
FROM <lookup table>
WHERE <key attribute> = '[LOOKUP]'
```

During execution of the query, the LOOKUP will be replaced by the current value of the Key Attribute of the current record and the SBN will be evaluated. If the Key Attribute is a character datatype, use quotes: `'[LOOKUP]'`. You can use functions to apply formatting or calculations:

```
SELECT DESTINATION
FROM LOOKUP_PRODUCTS
WHERE SOURCE = '[uRTrim(LOOKUP)]'
Optional Properties
Default Value
```

Optional properties

Default Value

Specify a Default Value to assign to the value attribute, in case the key value is not found in the lookup table.

Use Key Value

If Use Key Value is activated, the key value will be assigned to the value attribute instead of the default, if the lookup fails.

Lookup Empty/Null Keys

If activated, the lookup is performed even for empty or NULL key values. Otherwise the selected default method applies.

See also

Refer to the "Entering database connection parameters" on page 68 for more information on the previous and the following properties:

- Database
- Schema

- Standardize Data Format
- Database Options

Example

For 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 component therefore contains those three attributes.

The table to lookup the US 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, choose:

- Key Attribute: IN.PR_NUMMER
- Value Attribute: IN.PR_NUMMER
- Default Value: ‘INVALID’
- Query: SELECT DESTINATION FROM LOOKUP_PRODUCTS, where SOURCE = ‘[LOOKUP]’

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see *Demo DB Lookup Dynamic* in DemoRepository.

Staging components

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 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.

Use this component to stage transformation data in a dedicated area, perform aggregation, or create joins on data from heterogeneous sources.

Adding DB Staging to a project

Adding DB Staging to a project opens a property window. Use the property window to define connection parameters, staging tables, and query values.

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.

You could also use this component for creating an intermediate image of the transformation for further inspection or processing.

❖ **To use the staging component**

1 Add the component to the project and connect the ports to adjacent components.

2 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 Enter the Connection Parameters as described in “Entering database connection parameters” on page 68.

4 Create the staging tables. If the staging tables you are going to use already exist, go to step 6.

5 Right-click the component and select either Create Staging Table from Input or Create Staging Table from Port. The commands let you create the staging tables optionally based on the IN-port structure of the component or the structure of any other port within the project. However, you can also create the staging tables manually by using third party tools.

   Staging options are defined by using the Stage Options window. This window also lets you define the Truncate Table and the Write Block Size for each staging table.

6 Open the Query window by clicking Query button, and enter the query that selects the data from the staging area.

Optional properties

Read Block Size

The number of records forwarded to the next component within a single step is based on the value of Read Block Size.

Pre-processing SQL

One or more SQL statements to be executed during the initialization of the component. If you are using multiple statements, make sure you are separating them with a semicolon (;).

Post-processing SQL

One or more SQL statements to be executed after all components have finished execution. If you are using multiple statements, make sure you are separating them with a semicolon (;).

See “Entering database connection parameters” on page 68 for more information on the previous and the following properties:

- Database

- Schema
- Standardize Data Format
- Database Options

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 they act as a new data source for subsequent components. Both components allow overwriting of Read Block Size value of the original source components.

See also

For more information, see *Demo DB Staging* in DemoRepository.

Processing components

Component	Description
Executor JavaScript	Executor JavaScript is a general purpose component with no dedicated transformation functionality between its IN- and OUT-Port structures. You can use JavaScript to set up a procedure while transforming data. Use this component for the most flexible control about the data transformations applied to the records passed to the next component.

Executor JavaScript

Executor JavaScript is a general purpose component with no dedicated transformation functionality between its IN- and OUT-Port structures. You can use JavaScript to set up a procedure while transforming data.

Use this component for the most flexible control about the data transformations applied to the records passed to the next component.

Adding Executor JavaScript to a project

Add the component to the project and connect the ports to adjacent components.

Required properties

Script

Open the JavaScript Editor window to enter a JavaScript procedure describing the transformation process between the IN-Port and the OUT-Port.

Optional properties

Process Block

If this option is not activated, the transformation procedure is called on a record-by-record basis. This will cause the procedure to be executed for every incoming record. Activate this option to call the procedure once for every incoming whole block of records contained in the IN-port at the time of component step. In this case you are provided with two additional variables representing the IN-Port and OUT-Port content.

When processing in block mode, a unified approach can be used to transfer records form the input buffer IN to the output buffer OUT.

In addition to the buffer areas IN and OUT where the actual data are sitting, the Variables section also includes the IN.cols, IN.rows, OUT.cols and OUT.rows variables.

IN.cols and IN.rows contain the dimension of the input buffer (number of rows, number of columns).

Impact on simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Executor (Row mode)
- Demo XML via XSLT Data Provider (Block mode)

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.
DB Data Sink Delete	DB Data Sink Delete removes records from the 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. Use this component to delete records from the destination table.
DB Data Sink Synchronize	Updates or inserts records based on a selected key attributes values.
DB Data Sink Merge	Uses key attribute values to update and insert records in a destination table.
Text Data Sink	Writes transformation results to a text file in a delimited or fixed-length format.
XML Data Sink	Generates XML output from a data stream and writes the results to the local file system.

DB Data Sink Insert

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.

Adding DB Data Sink Insert to a project

Adding DB Data Sink Insert to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ To use DB Data Sink Insert

1. Add the component to the project and connect the ports to adjacent components.
2. Enter the Connection Parameters as described in “Entering database connection parameters” on page 68.
3. Create a destination table. If the destination table you are going to use already exists, continue with step 5.
4. Right-click the component and select either Add Destination Table from Input, or Add Destination Table from Port. Optionally, the command lets you create the destination table based on the IN-port structure of the component or the structure of any other port within the project. However, you might also create the destination table with your own toolset.
5. Click Destination Table, and select the table to be used from the list of available tables.

Optional properties

Insert Options

Click Insert Options to open the Insert Options window. By default, all attributes will be inserted. Deselect the attributes you want to exclude from insertion. This is useful if, for example, an attribute is automatically defaulted by the database system. You may specify a SQL INSERT value clause for each attribute, overwriting the incoming value of the selected attribute.

An example of the SQL INSERT value clause is:

```
'valid'
'[uDate("now")]'
```

Truncate Table

Activate this option to remove all records from the destination table when initializing the transformation process.

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the component.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins attribute names in SQL statements.

Closing Attribute Quote

The closing quote ends the attribute names in SQL statements.

Note If a record to be inserted violates any restrictions (for example: constraints, referential integrity, unique index definition) of the underlying table or object, an error message will occur. Rejected records are written to a log file if you provide the reject log DB Options. When executing a project from within a job the Continue on DB Write Errors option can be set to ignore rejection errors.

See "Entering database connection parameters" on page 68 for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Transfer German Customers
- Demo Transfer German Products
- Demo Transfer German Sales
- Demo Transfer US Customers
- Demo Transfer US Products

DB Data Sink Update

DB Data Sink Update 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 (for example: constraints, referential integrity, unique index definition), an error message occurs. The selected Key Value attribute is independent of any existing index definitions.

Adding DB Data Sink Update to a project

Adding DB Data Sink Update to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use the DB Data Sink Update component**

1 Add the component to the project and connect the ports to adjacent components.

2 Enter the Connection Parameters as described in “Entering database connection parameters” on page 68

3 Create a destination table. If the destination table you are going to use already exists, continue with step 5.

4 Right-click the component and select either Add Destination Table from Input or Add Destination Table from Port. Optionally, the command lets you create the destination table based on the IN-port structure of the component or the structure of any other port within the project. However, you might also create the destination table with your own toolset.

5 Click Destination Table and select the table to be used from the list of available tables.

6 Select one or more attributes composing the key. This key will be used to identify the records to be updated in the destination table. This is a logical selection, not related to any underlying indexes in the database schema.

Optional properties

Update options

Click Update Options to open the Update Options window. By default all attributes (key attributes are not listed) are selected. Deselect 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.

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See “Entering database connection parameters” on page 68 for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence.

There is no impact on the simulation sequence.

See also

For more information, see *Demo DB Datasink Update* in DemoRepository.

DB Data Sink Delete

DB Data Sink Delete removes records from the 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.

Use this component to delete records from the destination table.

Adding DB Data Sink Delete to a project

Adding DB Data Sink Delete to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use DB Data Sink Delete**

1 Add the component to the project and connect the ports to adjacent components.

2 Enter the Connection Parameters as described in “Entering database connection parameters” on page 68.

3 Create a destination table. If the destination table you are going to use already exist, continue with step 5.

4 Right-click the component and select either Add Destination Table from Input or Add Destination Table from Port. Optionally, the command lets you create the destination table based on the IN-port structure of the component or the structure of any other port within the project. However, you might also create the destination table with your own toolset.

5 Click Destination Table and select the table to be used from the list of available tables.

6 Select one or more attributes composing the key. This key is used to identify the records to be removed from the destination table. This is a logical selection, not related to any underlying indexes in the database schema.

Optional properties

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See section "Entering database connection parameters" on page 68 for more information on the previous properties and the following properties:

- Database
- Schema

- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see *Demo DB Datasink Delete* in DemoRepository.

DB Data Sink Synchronize

DB Data Sink Synchronize updates or inserts records based on a selected key value, which can consists of one or more attributes. DB Data Sink Synchronize uses selected key attributes of the incoming records to identify and process records in the destination table:

- If no matching record exists, DB Data Sink Synchronize inserts the input record.
- If matching records exists, DB Data Sink Synchronize updates the records with the incoming value.

Use this component to synchronize source and destination data. You can, for example, update existing records with current source values and insert new source records into the destination table.

Adding DB Data Sink Synchronize to a project

Adding DB Data Sink Synchronize to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use this component**

1. Add the component to the project and connect the ports to adjacent components.
2. Enter the Connection Parameters as described in the section Entering Database Connection Parameters.
3. Click the Destination Table button and select the table to be used from the list of available tables.

4 Select one or more attributes composing the key. This key will be used to identify the records to be updated in the destination table. If no matching record exists a new record will be inserted.

❖ **To enter optional properties**

1 Click the Update Options button to open the Update Options dialog Window. The Update Options apply to all records that are being updated. They do not apply to records that are inserted.

2 Deselect the attributes you want to exclude from update.

3 In the SQL UPDATE SET column specify an expression or constant that will be assigned to the record during the update (optional).

In a 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 comparison operator or expression allowed in the SQL language of the underlying database. Dynamic expression in square brackets will be evaluated during initialization of the component.

4 Click the Insert Options button to open the Insert Options dialog window. The Insert Options apply to all records that are being inserted. They do not apply to records that are updated.

A record with an identifier that does not exist in the table will be inserted with all the key attribute values plus the attributes selected in the Insert Options dialog window. Select or deselect any of the attributes. You can specify values for attributes that will be inserted, thus overwriting the corresponding value of the incoming attribute.

In the example above, all records that are being inserted will be inserted with the current date in its CU_CREATEDATE attribute.

Optional properties

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See section "Entering database connection parameters" on page 68for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see *Demo DB Datasink Synchronize* in DemoRepository.

DB Data Sink Merge

DB Data Sink Merge uses key attribute values to update and insert records in a destination table according to user defined rules.

Use this component for dimension tables that store current and historical data. You can, for example, use DB Data Sink Merge to insert a new instance of a particular record type.

Adding DB Data Sink Merge to a project

Adding DB Data Sink Merge to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use this component**

1. Add the component to the project and connect the ports to adjacent components.
2. Enter the Connection Parameters as described in the section Entering Database Connection Parameters.
3. Click the Destination Table button and select the table to be used from the list of available tables.
4. Click the Key button to open the Select Item dialog box.
5. Select one or more attributes composing the Key. The selected attributes are independent of any existing index definitions in the underlying database schema.
6. Click the Update Options button to open the Update Options dialog window. Select the attributes you want to update on existing records and enter the value you want to assign. You can specify an additional filter on the records with matching keys.

Example

You want to mark currently valid PRODUCTS records in the destination table that match the key attribute values of the incoming records, as invalid. To set up the update options for the above case, drag the PR_DESC attribute from the left window section to the right. Specify the condition and the value to assign, if that condition turns true. You can use any comparison operator or expression allowed in the SQL language of the underlying database.

In a SQL language notation the impact of the settings will read like:

```
UPDATE products
SET pr_desc = 'invalid'
WHERE pr_desc = 'valid'
AND <IN.attributes> =
<PRODUCTS.attributes>
```

Evaluation of an expression in square brackets is taking place during initialization of the component.

Optional properties

Insert Options

Click the Insert Options button to open the Insert Options window. By default all attributes will be inserted. Deselect the attributes you want to exclude from insertion. This is useful if, for example, an attribute is automatically defaulted by the database system. You may specify an SQL INSERT value clause for each attribute, overwriting the incoming value of the selected attribute.

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See section “Entering database connection parameters” on page 68for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information. see *Demo DB Datasink Merge* in DemoRepository.

DB Data Sink Synchronize

DB Data Sink Synchronize updates or inserts records based on a selected key value, which can consists of one or more attributes. DB Data Sink Synchronize uses selected key attributes of the incoming records to identify and process records in the destination table:

- If no matching record exists, DB Data Sink Synchronize inserts the input record.
- If matching records exists, DB Data Sink Synchronize updates the records with the incoming value.

Use this component to synchronize source and destination data. You can, for example, update existing records with current source values and insert new source records into the destination table.

Adding DB Data Sink Synchronize to a project

Adding DB Data Sink Synchronize to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use this component**

1. Add the component to the project and connect the ports to adjacent components.
2. Enter the Connection Parameters as described in the section Entering Database Connection Parameters.
3. Click the Destination Table button and select the table to be used from the list of available tables.
4. Select one or more attributes composing the key. This key will be used to identify the records to be updated in the destination table. If no matching record exists a new record will be inserted.

❖ **To enter optional properties**

1 Click the Update Options button to open the Update Options dialog Window. The Update Options apply to all records that are being updated. They do not apply to records that are inserted.

2 Deselect the attributes you want to exclude from update.

3 In the SQL UPDATE SET column specify an expression or constant that will be assigned to the record during the update (optional).

In a 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 comparison operator or expression allowed in the SQL language of the underlying database. Dynamic expression in square brackets will be evaluated during initialization of the component.

4 Click the Insert Options button to open the Insert Options dialog window. The Insert Options apply to all records that are being inserted. They do not apply to records that are updated.

A record with an identifier that does not exist in the table will be inserted with all the key attribute values plus the attributes selected in the Insert Options dialog window. Select or deselect any of the attributes. You can specify values for attributes that will be inserted, thus overwriting the corresponding value of the incoming attribute.

In the example above, all records that are being inserted will be inserted with the current date in its CU_CREATEDATE attribute.

Optional properties

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See section "Entering database connection parameters" on page 68for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information, see *Demo DB Datasink Synchronize* in DemoRepository.

DB Data Sink Merge

DB Data Sink Merge uses key attribute values to update and insert records in a destination table according to user defined rules.

Use this component for dimension tables that store current and historical data. You can, for example, use DB Data Sink Merge to insert a new instance of a particular record type.

Adding DB Data Sink Merge to a project

Adding DB Data Sink Merge to a project opens a configuration window and properties section. Use the configuration window and properties section to identify the connection parameters, destination table, and processing options.

❖ **To use this component**

1. Add the component to the project and connect the ports to adjacent components.
2. Enter the Connection Parameters as described in the section Entering Database Connection Parameters.
3. Click the Destination Table button and select the table to be used from the list of available tables.
4. Click the Key button to open the Select Item dialog box.
5. Select one or more attributes composing the Key. The selected attributes are independent of any existing index definitions in the underlying database schema.
6. Click the Update Options button to open the Update Options dialog window. Select the attributes you want to update on existing records and enter the value you want to assign. You can specify an additional filter on the records with matching keys.

Example

You want to mark currently valid PRODUCTS records in the destination table that match the key attribute values of the incoming records, as invalid. To set up the update options for the above case, drag the PR_DESC attribute from the left window section to the right. Specify the condition and the value to assign, if that condition turns true. You can use any comparison operator or expression allowed in the SQL language of the underlying database.

In a SQL language notation the impact of the settings will read like:

```
UPDATE products
SET pr_desc = 'invalid'
WHERE pr_desc = 'valid'
AND <IN.attributes> =
<PRODUCTS.attributes>
```

Evaluation of an expression in square brackets is taking place during initialization of the component.

Optional properties

Insert Options

Click the Insert Options button to open the Insert Options window. By default all attributes will be inserted. Deselect the attributes you want to exclude from insertion. This is useful if, for example, an attribute is automatically defaulted by the database system. You may specify an SQL INSERT value clause for each attribute, overwriting the incoming value of the selected attribute.

Write Block Size

The Write Block Size defines the number of records to be written to the database in a single write operation.

Pre-processing SQL

This property provides one or more SQL statements to be executed during initialization of the project.

Post-processing SQL

This property provides one or more SQL statements to be executed when all components finished execution.

Opening Attribute Quote

The open quote character begins the attribute names in SQL statements.

Closing Attribute Quote

The close quote character ends the attribute names in SQL statements.

See section "Entering database connection parameters" on page 68for more information on the previous properties and the following properties:

- Database
- Schema
- Standardize Data Format
- Database Options

Impact on the simulation sequence

There is no impact on the simulation sequence.

See also

For more information. see *Demo DB Datasink Merge* in DemoRepository.

Text Data Sink

Text Data Sink writes transformation results to a text file in a delimited or fixed-length format.

Adding Text Data Sink to a project

Adding Text Data Sink to a project opens a component window and properties section. Use the configuration window and properties section to identify the port properties, destination file, and processing options. Click Properties to open the component window from the properties section.

❖ **To add a Text Data Sink to a project**

1. Add the component.
2. Enter a file name for the destination file.
3. Describe the file structure: Type, Line Delimiter, Column Delimiter.
4. Click Save to exit the component window.
5. Connect the component with the OUT-Port of an adjacent component.
6. Click Open to open the Text Data Sink component window and verify the result. You may add or remove columns and reassign the port structure.

Exporting and importing File Definitions files

Use the Export command in the Properties menu to save your current settings to a file definition file. Use the Import command in the Properties menu to load an existing file definition file that has been previously created with the Export command.

Modifying the port structure (delimited files)

When you modify the column structure within the Text Data Sink component window you can recreate the current port structure or assign a new port structure.

- To select a new port structure, click the Assign Port Structure button.
- To go back to the current IN-port structure, click the Regenerate Column Definition button.

Working with fixed-length files

After selecting one of the fixed-length file types, you must create the columns and provide the position parameters of each column.

❖ **To add columns to the output**

1 Click Open to open the Text Data Sink component window.

2 Click Insert a New Attribute. You may edit the name of the generated column.

❖ **To remove columns from the output**

1 Click the Open button to open the Text Data Sink component window.

2 Click Remove.

Properties

Character Encoding

Select a character set from the drop-down list and press Enter to confirm.

Type

Specify the type of the input file and press Enter to confirm.

Delimited

The fields in the file will be separated by a specific character or string.

Fixed (Fixed Line)

The fields and lines in the file are of fixed length. There is no line delimiter.

Fixed (Variable Line)

The fields in the file are of fixed length. The lines are separated by a specific character or string.

Line Delimiter

Either select from the list or enter a value, and then press Enter to confirm.

Line Length

Enter a number that matches the line length for a Fixed (Fixed Line) file and press Enter to confirm.

Column Delimiter

Either select from the list or enter a value, and then press Enter to confirm.

Write Column Names

The column names will be written into the first line of the file. Press Enter to confirm.

Quote characters

If the field values are quoted in the destination text file, provide the respective character. Press Enter to confirm.

Append

Activate this option to append the incoming data to an existing file.

Header

Enter text for a header to be written to the file before writing out the incoming data. Expressions are allowed in Square Bracket Notation.

Impact on simulation sequence

There is no impact on simulation sequence.

See also

For more information. see *Demo Text Data Sink Delimited/Fixed* in DemoRepository.

XML Data Sink

XML Data Sink generates XML output from a data stream and writes the results to the local file system.

The properties of the XML Data Sink can be changed either from within the XML Data Sink component window or by manipulating the items in the Property section. Use the Properties button to open the component window from the Property section.

An **attribute** is the equivalent of a Data Element (Field) on a Sybase ETL Small Business Edition data stream. It has a name and a (changing) value. There are two common ways to define Attribute Values in XML: as an Attribute or a Tag. You can select the way you want your attribute to show up in your XML Document by right clicking on an entry in the XML Output Definition and select Node Type.

A Node can either be a Name/Value combination or a Structure Node that refers to a substructure.

Adding XML Data Sink to a project

Adding XML Data Sink to a project opens a Component window and a properties section. Use the objects to define the structure, content, and location of the XML output.

❖ **To add an XML Data Sink Component to a project**

1 Add the component.

2 Enter a file name for the destination file by using the Property Window or selecting Destination from the Properties menu.

3 You can change the content and the shape of the XML Document by either dragging Input Port Attributes to the XML Output Definition or by right-clicking on any item listed in the XML Output Definition.

4 Exit the component window with the Save button.

Note The XML Write component writes the XML output in the order it receives from the incoming data stream. If you want to ensure that your hierarchies are represented correctly, you need make sure that the attributes used for grouping your hierarchies are delivered in the right order.

If the originating data stream does not deliver the data stream in the right order, you should consider using a DB Staging Component before outputting the data to the XML document. The SQL Statement delivering the data FROM the DB Staging Component should have an ORDER BY clause containing all attributes you use for grouping in your XML component in the right order.

Job components

Job components control job execution.

Component	Description
Start	Allows you to schedule a job and determine when that job begins. 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.
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. Before you add Finish to a job, your job must one contain one of the following job components: Synchronize, Project or a Multi-Project.
Error	Provides a visual representation of the end of a failed job execution. Use this component to mark the end of a failed job. Before you add Error to a job, your job must one contain one of the following job components: Synchronize, Project or a Multi-Project.

Start

Start allows you to schedule a job and determine when that job begins. Start is the first component you add to any job.

Adding the Start component to a job

Dragging the Start component to the Design section opens a properties section. Click the left option of the Start Time property or double-click the Start icon in the Design section to schedule the job.

❖ **To schedule a job**

1 Open the Create Schedule dialog box by clicking the left option of the Start Time property.

2 Enter Name, Description, Username, and Password of the system account that runs the job and confirm your settings.

3 In the Schedule window, enter the scheduling data.

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

Dragging the Project component to the Design section opens a properties section. Click the Project icon in the properties section, or double-click the Project icon in the Design section to open a Project properties sheet.

❖ **To use this component**

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

2 Double-click the component and select a project from the list of projects.

Note Multiple projects can be connected to the Start Component.

Optional properties

Continue on DB Write Errors

If you activate this option, project execution will 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 “Entering database connection parameters” on page 68) this option lets you “post-process” rejected records.

See also

View the following jobs in DemoDatastore:

- Demo Transfer all German Data
- Demo Transfer US 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 cause the Synchronizer to signal failure.

❖ **To use this component**

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

2 In the Property section click the Synchronize Options button to select critical projects (optional).

See also

For more information, see *Demo Transfer all German Data* in DemoDatastore:

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.

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.

❖ **To use this component**

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

2 Click option for the Projects Execution property to open the project selection window.

Project options

For each project to be executed the following options are available.

Continue on Error

This option corresponds to the Continue on DB Write Errors property of a Project component. If you activate 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.

- To add projects to the group, select the projects in the tree on the left-hand side and select Add Project(s) from the pop-up menu.
- To remove projects from the group, select the projects in the list (or in the tree) and select Remove Project(s) from the pop-up menu.

Note A project group can contain only one instance of a project. Trying to add a project multiple times will have no effect.

See also

For more information, see *Demo Transfer all US Data* in DemoDatastore:

Finish

Finish visually represents the end of a successful job execution. Use this component to mark the successful end of a job. Before you add Finish to a job, your job must one contain one of the following job components: Synchronize, Project or a Multi-Project.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Transfer all German Data

- Demo Transfer all US Data
- Demo Transfer US 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. Before you add Error to a job, your job must one contain one of the following job components: Synchronize, Project or a Multi-Project.

See also

For more information, see these component demonstrations in DemoRepository:

- Demo Transfer all German Data
- Demo Transfer all US Data

APPENDIX A

Function Reference

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

Note Even if your original data is not coded in UNICODE, and you have used one of the "u"-functions, the returned data has been converted to UNICODE and needs to be handled accordingly.

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 it's 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

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
```

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
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`	hundreths of seconds
`t`	thousandth of seconds
`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

Function	Description
Working with date and time functions	Describes working with date and time functions.
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
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

Working with date and time functions

Description

Describes working with date and time functions

Usage

Time Strings

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.

Most of the 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").

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

```
IN.Date = uDate("2004-01-04 14:26:33") // returns the
date part "2004-01-04"
```

Modifiers

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

14 julian

15 gregorian

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.

The julian modifier (14) assumes that the time string is a gregorian date and converts the date into a julian date: gregorian (15) undoes the work of julian.

Date and time calculations

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 days since the battle of Hastings

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

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

The computation of local time depends heavily on the whim of local politicians and is thus difficult to get correct for all locales. 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. When you use the julian modifier, it does not convert the date into a real Julian calendar date, it merely shifts the Gregorian calendar date to align it with the Julian calendar. This means that the julian modifier will not work right for dates that exist in the Julian calendar but which do not exist in the Gregorian calendar. Example: 1900-02-29.

uDate

Description

Returns a year, month and day from a date in the format *YYYY-MM-DD*

Note Refer to Working with date and time functions for detailed information about the possible modifier arguments.

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*

Note Refer to Working with date and time functions for detailed information about the possible modifier arguments.

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

Note Please refer to Working with date and time functions for detailed information about the possible modifier arguments.

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

Note Please refer to Working with date and time functions for detailed information about the possible modifier arguments.

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

Note Refer to Working with date and time functions for detailed information about the possible modifier arguments.

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.

Note Please refer to Working with date and time functions for detailed information about the possible modifier arguments.

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"
```

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

Note Please refer to Working with date and time functions for detailed information about the possible modifier arguments.

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

Note Refer to Working with date and time functions for detailed information about the possible modifier arguments.

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

Note Refer to Working with date and time functions for detailed information about the possible modifier arguments.

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	Set 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

Set 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 ETL Small Business Edition 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	Returns
%A	**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
%%	%

Note Please refer to Working with date and time functions for detailed information about the possible modifier arguments.

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
```

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 (i.e., 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 floting point number

```
uMatches("abc","[-+]?[0-9]*\.?[0-9]*")  // return 0
uMatches("1.23","[-+]?[0-9]*\.?[0-9]*")  // return 1
```

Lookup

Function	Description
uChoice	Return the value of a given parameter specified by an index
uFirstDifferent	Return the first parameter value which is different from the first parameter
uFirstNotNull	Returns the first non null parameter.
uElements	Return the number of elements in a delimited string
uToken	Return the Nth element from a delimited string

uChoice

Description

Return 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: Return 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: Return 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

Return 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	Set 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 something like
"D:\ETL\win32\Engine.exe engineman.lbr"
```

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"
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"

Primary Language	Sublanguage	Language String
Korean	Korean	"kor" or "korean"
Norwegian	Norwegian (Bokmal)	"nor" or "norwegianbokmal"
Norwegian	Norwegian (default)	"norwegian"
Norwegian	Norwegian (Nynorsk)	"non" or "norwegiannynorsk"
Polish	Polish	"plk" or "polish"
Portuguese	Portuguese (Brazil)	"portuguese-brazilian" or "ptb"
Portuguese	Portuguese (default)	"portuguese" or "ptg"
Russian	Russian (default)	"rus" or "russian"
Slovak	Slovak	"sky" or "slovak"
Spanish	Spanish (default)	"esp" or "spanish"
Spanish	Spanish (Mexican)	"esm" or "spanish-mexican"
Spanish	Spanish (Modern)	"esn" or "spanish-modern"
Swedish	Swedish	"sve" or "swedish"
Turkish	Turkish	"trk" or "turkish"

Country/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"

Country/Region	Country/Region String
Hong Kong SAR	"hkg", "hong kong", or "hong-kong"
Hungary	"hun" or "hungary"
Iceland	"iceland" or "isl"
Ireland	"ireland" or "irl"
Italy	"ita" or "italy"
Japan	"japan" or "jpn"
Korea	"kor", "korea"
Mexico	"mex" or "mexico"
Netherlands	"nld", "holland", or "netherlands"
New Zealand	"new zealand", "new-zealand", "nz", or "nzl"
Norway	"nor" or "norway"
People's Republic of China	"china", "chn", "pr china", or "pr-china"
Poland	"pol" or "poland"
Portugal	"prt" or "portugal"
Russia	"rus" or "russia"
Singapore	"sgp" or "singapore"
Slovak Repubic	"svk" or "slovak"
Spain	"esp" or "spain"
Sweden	"swe" or "sweden"
Switzerland	"che" or "switzerland"
Taiwan	"taiwan" or "twn"
Turkey	"tur" or "turkey"
United Kingdom	"britain", "england", "gbr", "great britain", "uk", "united kingdom", or "united-kingdom"
United States of America	"america", "united states", "unitedstates", "us", or "usa"

uSleep

Description — Suspends the process for a specified 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*
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

Group	Name	Description
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	BITBUCKET	The virtual folder containing the objects in the user's Recycle Bin.
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_ALTSTARTUP	The file system directory that corresponds to the nonlocalized Startup program group for all users. Valid only for Microsoft Windows NT systems.
Windows	COMMON_APPDATA	The file system directory containing application data for all users. A typical path is *C:\Documents and Settings\All Users\Application Data*.
Windows	COMMON_DESKTOPDIRECTORY	The file system directory that contains files and folders that appear on the desktop for all users. A typical path is *C:\Documents and Settings\All Users\Desktop*. Valid only for Windows NT systems.
Windows	COMMON_DOCUMENTS	The file system directory that contains documents that are common to all users. A typical paths is *C:\Documents and Settings\All Users\Documents*.
Windows	COMMON_FAVORITES	The file system directory that serves as a common repository for favorite items common to all users. Valid only for Windows NT systems.

Group	Name	Description
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.
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.

Group	Name	Description
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	DRIVES	The virtual folder representing My Computer, containing everything on the local computer: storage devices, printers, and Control Panel. The folder may also contain mapped network drives.
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	A viritual folder for Internet Explorer (icon on desktop).
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	LOCAL_APPDATA	The file system directory that serves as a data repository for local (nonroaming) applications. A typical path is *C:\Documents and Settings\username\Local Settings\Application Data*.
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	NETWORK	A virtual folder representing Network Neighborhood, the root of the network namespace hierarchy.
Windows	PERSONAL	The virtual folder representing the My Documents desktop item. This is equivalent to MYDOCUMENTS.
Windows	PRINTERS	The virtual folder containing installed printers.
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.

Group	Name	Description
Windows	SENDTO	The file system directory that contains Send To menu items. A typical path is *C:\Documents and Settings\username\SendTo*.
Windows	STARTMENU	The file system directory containing Start menu items. A typical path is *C:\Documents and Settings\username\Start Menu*.
Windows	STARTUP	The file system directory that corresponds to the user's Startup program group. The system starts these programs whenever any user logs 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 `BCC` 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
uCeil	Returns least integer greater than or equal to argument
uDiv	Return the division integer
uExp	Returns the exponential, base e
uFloor	Returns greatest integer less than or equal to argument
uLn	Return the natural logarithm (base e) of a number
uLog	Return the logarithm of a number
uMod	Return 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: Return 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 — Return 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 — Return 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: Return 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 of functions and procedures

```
uEvaluate("if (IN.CUSTNO > 22) {return 22;} else {return
IN.CUSTNO;} )") // Return IN.CUSTNO if IN.CUSTNO is
below 22 or return 22 if above.
```

You can also define your functions for later use. If you evaluate the following:

```
function myFunc(a ,b)
{ return a+b;
}
```

Now you can call your function using uEvaluate

```
uEvaluate("myFunc(3, 5)") // Returns 8
```

String

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

Sybase ETL Server

The Sybase ETL Server is the execution component of Sybase ETL Small Business Edition product suite. This appendix describes usage and architecture of the ETL Server application.

The following terms are used:

Sybase

- ETL Server — The server application itself, which provides several independent services.
- GRID engine — Part of the server application, which actually executes jobs or projects.

GRID

A GRID engine running as a server is waiting for execution requests through TCP/IP. The default port is 5124 and can be customized.

Note Verify that you have no firewall blocking this port or change the port on to a different number better matching your environment.

Troubleshooting

❖ **To prepare the support contact**

1 Check the error text.

2 Check the log file.

3 Run again with system trace switched on.

4 Check the version and revision number as well as your machine identification.

Syntax:

```
GridNode --version
```

Output:

```
Sybase ETL Small Business Edition (4.1.0.676)
Copyright © Sybase, Inc. 2002-2006
Grid Node:
Machine ID: 9TuH/ioF6Wt/Gig=
```

5 Check the licenses available.

Syntax:

```
GridNode --licenses
```

Output:

```
Sybase ETL Small Business Edition (4.1.0.676)
Copyright (c) Sybase, Inc. 2002-2006
Grid Node:
Product ID: Sybase ETL Small Business Edition
Machine ID: 9TuH/ioF6Wt/Gig=
File: custom
```

```
Product: Sybase ETL Small Business Edition
Version: 4.1
License: Enterprise Edition
Status: Valid
Expiration:
```

INI file settings

In the *etc* directory of the product installation you will find INI files which you can use in order to setup some preferences permanently.

Every application is looking for its own INI file named *[application name].ini*. If a certain key is not found, the *Default.ini* file will be searched.

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.
Language	Default	English_USA	English_USA	
Logging	Default	1/0	0	
Logging	File	1/0	1	
Logging	Tracelevel	0-5	0	
Logging	Flushtime	1-n	1	Indicates seconds between the internal log flashes.

GridNode.ini

Group	Key	Values	Default	Description
Network	GridDefaultPort	1-65536	5124	The default GridNode port

Group	Key	Values	Default	Description
Network	BroadcastAddress	Any IP address	255.255.255.255	The broadcast address for GridNode UDP messages
Identification	NodeDomain	any	Global	(not used yet)

APPENDIX C

Queuing and Executing Process Calls

ProcessQ is an application that can queue and execute process calls parallel or in sequence. The serialization is implemented with a mutex (mutual exclusion) semaphore. The application uses an internal GUID to name the mutex, so the exclusion is system-wide (over all products). It is possible to set an own ID for special purposes. Both the state of being locked by the mutex and the runtime of the queued process have an own timeout mechanism. The windows style (hidden, normal, maximized, minimized) of the application to call can be configured.

ProcessQ requires the SolBase.dll library.

Configuring ProcessQ calls

ProcessQ.exe can be configured with several parameters. Only -E is required; all other parameters are optional.

-S

Serializes the process calls. The default is parallel execution.

-TP n

Timeout, the number of milliseconds before the RUNNING process will stop. By default, no timeout occurs.

-TL n

Timeout, the number of milliseconds before the LOCKED process will stop. By default, no timeout occurs.

-I

Customized ID for exclusion.

-W n

Window style; see “Controlling the appearance of a new process” on page 235.

-D

The working directory for the application.

-E

The executable and its command line parameters. If this value contains spaces, it must be quoted with double quotes.

-P<ExtVar=Value>[,< ExtVar=Value>..,< ExtVar=Value>]

This option allows you to set parameters from the command line that can be used by the job or project during execution. <ExtVar> is an environment variable that you can access in a Sybase ETL Small Business Edition object during execution with the uGetEnv() function. Multiple variables are separated by a comma, the full string should be quoted using double quotes. Spaces within a value can be quoted with single quotes. Spaces within the key are not allowed.

Examples

o processq -P "HOME=h:\" -E emacs.exe

o processq -W 0 -S -P Port=8080 -E "d:\tod\engine.exe d:\tod\engineman.lbr"

o processq -P "HOME='h:\BS test 1\BS test'; LANG=de" -E transform.exe

Errorcodes

All error messages during execution are written into a log file. The name of the file will depend on the module name which raises the error, in this case *ProcessQ.log*. The path is the current working directory and can be set by the environment variable SOLONDE LOG DIR.

Code	Error
1	Invalid usage of command line. This also displays the available options.
2	Error when parsing the command line.
3	Error setting the environment variable.
4	No executable given.
5	Creating mutex failed.
6	Mutex timed out.
7	Creating process (running the executable) failed.
8	Terminating the process failed.

Controlling the appearance of a new process

By providing a Window Style with the -W option, you can control the appearance of the new process.

Value	Name	Meaning
0	HIDE	Hides the window and activates another window.
1	SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when displaying the window for the first time.
2	SHOWMINIMIZED	Activates the window and displays it as a minimized window.
3	SHOWMAXIMIZED	Maximizes the specified window.
4	SHOWNOACTIVATE	Displays a window in its most recent size and position. This value is similar to SHOWNORMAL, except the window is not active.

Value	Name	Meaning
5	SHOW	Activates the window and displays it in its current size and position.
6	MINIMIZE	Minimizes the specified window and activates the next top-level window in the Z order.
7	SHOWMINNOACTIVE	Displays the window as a minimized window. This value is similar to SHOWMINIMIZED, except the window is not activated.
8	SHOWNA	Displays the window in its current size and position.
9	RESTORE	Activates and displays the window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when restoring a minimized window.
10	SHOWDEFAULT	Sets the show state based on the value specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application.
11	FORCEMINIMIZE	Windows 2000/XP: Minimizes a window, even if the thread that owns the window is hanging. This flag should be used only when minimizing windows from a different thread.

Index

E

F

G

I

J

L

M

N

O

P

R

S

T

U

X