TIBCO® Object Service Broker UI

Help

Software Release 6.0
July 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, The Power of Now, TIBCO Object Service Broker, and and TIBCO Service Gateway are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

The TIBCO Object Service Broker technologies described herein are protected under the following patent numbers:

Australia:	-	-	671137	671138	673682	646408
Canada:	2284250	-	-	2284245	2284248	2066724
Europe:	-	-	0588446	0588445	0588447	0489861
Japan:	-	-	-	-	-	2-513420
USA:	5584026	5586329	5586330	5594899	5596752	5682535

Contents

Preface

TIBCO Object Service Broker provides an application development environment that allows you to create applications that integrate various systems in your enterprise.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Object Service Broker Documentation

The following documents form the TIBCO Object Service Broker documentation set:

Fundamental Information

The following manuals provide fundamental information about TIBCO Object Service Broker:

- *TIBCO Object Service Broker Getting Started* Provides the basic concepts and principles of TIBCO Object Service Broker and introduces its components and capabilities. It also describes how to use the default developer's workbench and includes a basic tutorial of how to build an application using the product. A product glossary is also included in the manual.
- *TIBCO Object Service Broker Messages with Identifiers* Provides a listing of the TIBCO Object Service Broker messages that are issued with alphanumeric identifiers. The description of each message includes the source and explanation of the message and recommended action to take.
- *TIBCO Object Service Broker Messages without Identifiers* Provides a listing of the TIBCO Object Service Broker messages that are issued without a message identifier. These messages use the percent symbol (%) or the number symbol (#) to represent such variable information as a rules name or the number of occurrences in a table. The description of each message includes the source and explanation of the message and recommended action to take.
- *TIBCO Object Service Broker Quick Reference* Presents summary information for use in the TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Shareable Tools* Lists and describes the TIBCO Object Service Broker shareable tools. Shareable tools are programs supplied with TIBCO Object Service Broker that facilitate rules language programming and application development.
- *TIBCO Object Service Broker Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Application Development and Management

The following manuals provide information about application development and management:

- *TIBCO Object Service Broker Application Administration* Provides information required to administer the TIBCO Object Service Broker application development environment. It describes how to use the administrator's workbench, set up the development environment, and optimize access to the database. It also describes how to manage the Pagestore, which is the native TIBCO Object Service Broker data store.
- *TIBCO Object Service Broker Managing Data* Describes how to define, manipulate, and manage data required for a TIBCO Object Service Broker application.
- *TIBCO Object Service Broker Managing External Data* Describes the TIBCO Object Service Broker interface to external files (not data in external databases) and describes how to define TIBCO Object Service Broker tables based on these files and how to access their data.
- *TIBCO Object Service Broker National Language Support* Provides information about implementing the National Language Support in a TIBCO Object Service Broker environment.
- *TIBCO Object Service Broker Object Integration Gateway* Provides information about installing and using the Object Integration Gateway which is the interface for TIBCO Object Service Broker to XML, J2EE, .NET and COM.
- *TIBCO Object Service Broker for Open Systems External Environments* Provides information on interfacing TIBCO Object Service Broker with the Windows and Solaris environments. It includes how to use SDK (C/C++) and SDK (Java) to access TIBCO Object Service Broker data, how to interface to TIBCO Enterprise Messaging Service (EMS), how to use the TIBCO Service Gateway for WMQ, how to use the Adapter for JDBC-ODBC, and how to access programs written in external programming languages from within TIBCO Object Service Broker.
- *TIBCO Object Service Broker for z/OS External Environments* Provides information on interfacing TIBCO Object Service Broker to various external environments within a TIBCO Object Service Broker z/OS environment. It also includes information on how to access TIBCO Object Service Broker from different terminal managers, how to write programs in external programming languages to access TIBCO Object Service Broker data, how to interface to TIBCO Enterprise Messaging Service (EMS), how to use the TIBCO Service Gateway for WMQ, and how to access programs written in external programming languages from within TIBCO Object Service Broker.

- *TIBCO Object Service Broker Parameters* Lists the TIBCO Object Service Broker Execution Environment and Data Object Broker parameters and describes their usage.
- *TIBCO Object Service Broker Programming in Rules* Explains how to use the TIBCO Object Service Broker rules language to create and modify application code. The rules language is the programming language used to access the TIBCO Object Service Broker database and create applications. The manual also explains how to edit, execute, and debug rules.
- *TIBCO Object Service Broker Managing Deployment* Describes how to submit, maintain, and manage promotion requests in the TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Defining Reports* Explains how to create both simple and complex reports using the reporting tools provided with TIBCO Object Service Broker. It explains how to create reports with simple features using the Report Generator and how to create reports with more complex features using the Report Definer.
- *TIBCO Object Service Broker Managing Security* Describes how to set up, use, and administer the security required for an TIBCO Object Service Broker application development environment.
- *TIBCO Object Service Broker Defining Screens and Menus* Provides the basic information to define screens, screen tables, and menus using TIBCO Object Service Broker facilities.
- *TIBCO Service Gateway for Files SDK* Describes how to use the SDK provided with the TIBCO Service Gateway for Files to create applications to access Adabas, CA Datacom, and VSAM LDS data.

System Administration on the z/OS Platform

The following manuals describe system administration on the z/OS platform:

- *TIBCO Object Service Broker for z/OS Installing and Operating* Describes how to install, migrate, update, maintain, and operate TIBCO Object Service Broker in a z/OS environment. It also describes the Execution Environment and Data Object Broker parameters used by TIBCO Object Service Broker.
- *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* Explains the backup and recovery features of OSB for z/OS. It describes the key components of TIBCO Object Service Broker systems and describes how you can back up your data and recover from errors. You can use this information, along with assistance from TIBCO Support, to develop the best customized solution for your unique backup and recovery requirements.

- *TIBCO Object Service Broker for z/OS Monitoring Performance* Explains how to obtain and analyze performance statistics using TIBCO Object Service Broker tools and SMF records
- *TIBCO Object Service Broker for z/OS Utilities* Contains an alphabetically ordered listing of TIBCO Object Service Broker utilities for z/OS systems. These are TIBCO Object Service Broker administrator utilities that are typically run with JCL.

System Administration on Open Systems

The following manuals describe system administration on open systems such as Windows or UNIX:

- *TIBCO Object Service Broker for Open Systems Installing and Operating* Describes how to install, migrate, update, maintain, and operate TIBCO Object Service Broker in Windows and Solaris environments.
- *TIBCO Object Service Broker for Open Systems Managing Backup and Recovery* Explains the backup and recovery features of TIBCO Object Service Broker for Open Systems. It describes the key components of a TIBCO Object Service Broker system and describes how to back up your data and recover from errors. Use this information to develop a customized solution for your unique backup and recovery requirements.
- *TIBCO Object Service Broker for Open Systems Utilities* Contains an alphabetically ordered listing of TIBCO Object Service Broker utilities for Windows and Solaris systems. These TIBCO Object Service Broker administrator utilities are typically executed from the command line.

External Database Gateways

The following manuals describe external database gateways:

- *TIBCO Service Gateway for DB2 Installing and Operating* Describes the TIBCO Object Service Broker interface to DB2 data. Using this interface, you can access external DB2 data and define TIBCO Object Service Broker tables based on this data.
- *TIBCO Service Gateway for IDMS/DB Installing and Operating* Describes the TIBCO Object Service Broker interface to CA-IDMS data. Using this interface, you can access external CA-IDMS data and define TIBCO Object Service Broker tables based on this data.
- *TIBCO Service Gateway for IMS/DB Installing and Operating* Describes the TIBCO Object Service Broker interface to IMS/DB and DB2 data. Using this interface, you can access external IMS data and define TIBCO Object Service Broker tables based on it.

- *TIBCO Service Gateway for ODBC and for Oracle Installing and Operating*
  Describes the TIBCO Object Service Broker ODBC Gateway and the TIBCO Object Service Broker Oracle Gateway interfaces to external DBMS data. Using this interface, you can access external DBMS data and define TIBCO Object Service Broker tables based on this data.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *OSB_HOME*	By default, all TIBCO products are installed into a folder referenced in the documentation as *TIBCO_HOME*. On open systems, TIBCO Object Service Broker installs by default into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *OSB_HOME*. The default value of *OSB_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\OSB`. Similarly, all TIBCO Service Gateways on open systems install by default into a directory in *TIBCO_HOME*. For example on Windows systems, the default value is `C:\tibco\OSBgateways\6.0`. On z/OS, no default installation directories exist.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

Table 2 Syntax Typographical Conventions

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts, a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access All TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Views

This chapter describes the views provided by the TIBCO Object Service Broker UI.

Topics

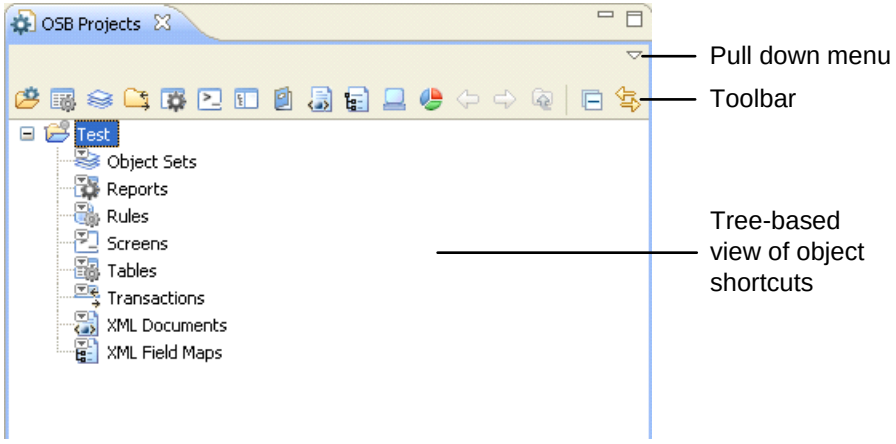
OSB Projects View

Using OSB Projects View

The OSB Projects view serves two purposes. First, it defines a connection to a particular TIBCO Object Service Broker server. The connection parameters include host name, port number, user id, password, and other settings.

Secondly, you can use an OSB Projects to store shortcuts to particular objects on the TIBCO Object Service Broker server. These shortcuts are added manually from various places in the UI and can be removed from the project without deleting the actual objects.

OSB projects are stored in the Eclipse workspace; you can define any number of OSB projects in a single workspace. The OSB Projects view displays OSB projects in a tree view:



You can open other views by clicking the icons on the toolbar:

Icon	Description
1	Opens the Open Object Dialog
2	Opens the Tables View.

Icon	Description
3	Opens the Object Sets View.
4	Opens the Applications View.
5	Opens the Reports View.
6	Opens the Screens View.
7	Opens the Libraries View.
8	Opens the Transaction Logs View.
9	Opens the XML Documents View.
10	Opens the XML Field Maps View.
11	Opens the TIBCO Object Service Broker text workbench.
12	Opens the Profiler View.
13 - 17	Refer to the Eclipse package explorer help.

You can also open these views via nodes in the tree structure. For example, to open XML Documents view, click the icon on the left of the XML Documents node and select Show XML Documents.

For information on the toolbar pull down menu, refer to the Eclipse Package Explorer help.

OSB Projects versus Other Views

OSB Projects view has a special place in the UI. It is the only view that shows the object shortcuts added to the OSB project. All other views show the objects present on the server. In addition, each view (except OSB Projects) is associated with a particular project.

OSB Project view acts as a central point for opening other views. For example, if you press the **Icon 2** button listed in the table above, the UI will pick up the currently selected project and display the view of tables on the respective server.

In order to see which project an editor is associated with, place your mouse on the editor. This will show a tooltip with the project name.

Applications View

The Applications view displays a table that lists the applications defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Column	Description
Name	Name of the application.
Unit	Name for a group of related objects.
Title	Title of the application.
Description	Description of the application.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the applications that appear in the table. For details, see View Filter and Refresh.

Working with Applications

To perform a task on an application, locate the desired application in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired application.
- Right-click the text in the Name column of the desired application.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected application in the Application editor.
Delete	Deletes the selected application. This option is valid for multiple selections.

Task Selection	Description
Show Transactions	Displays the transactions in the selected application in the Transactions view

Libraries View

The Libraries view displays a table that lists the libraries defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Column	Description
Name	Name of the library.
Unit	Name for a group of related objects.
Modified	Date the library was last modified.
Modifier	Id of the user who last modified the library.
Summary	Explanation of the library.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the libraries that appear in the table. For details, see View Filter and Refresh.

Working with Libraries

To perform a task on an library, locate the desired library in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired library.
- Right-click the text in the Name column of the desired library.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected library in the Library editor.
Delete	Deletes the selected library. This option is valid for multiple selections.

Task Selection	Description
Rename	Renames the selected library. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected library into the system clipboard. This option is valid for multiple selections.
Paste	Paste a library you have copied. When a prompt appears, type a new name for the library you are pasting in the **New name** field. You can include all the rules contained in the library you are pasting by checking Copy rules.
Show Rules	Displays the rules in the selected library in the Rules view.

Transaction Logs View

The Transaction Logs view displays a table that lists the transaction logs on the TIBCO Object Service Broker server. The table consists of the following columns:

Column	Description
Time	Time the transaction log was created.
Data	Contents of a line from the transaction log.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the transaction logs that appear in the table. For details, see View Filter and Refresh.

Deleting

The task menu has a Delete selection for deleting the selected transaction logs.

Object Sets View

The Object Sets view displays a table that lists the object sets defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the object set.
Unit	Name for a group of related objects.
Created	Date the object set was created.
Author	Id of the user who created the object set.
Modified	Date the object set was last modified.
Modifier	Id of the user who last modified the object set.
Summary	Explanation of the object set.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the object sets that appear in the table. For details, see View Filter and Refresh.

Working with Object Sets

To perform a task on an object set, locate the desired object set in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired object set.
- Right-click the text in the Name column of the desired object set.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected object set in the Object Set editor.
Add to Project…	Adds the selected object set to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected object set into the system clipboard. This option is valid for multiple selections.
Rename	Renames the selected object set. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected object set into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the object set you have copied. When a prompt appears, type a new name for the object set you are pasting in the **New name** field.

Profiler View

The Profiler view displays information about executing rules (profile) with the intent to optimize performance. There are two tabs on this view – Tree and Totals. The Profiler Result Selection Dialog window is used to select a previous profiler result; the Dialog window can be invoked using the Filter button in the Profiler view.

Tree Tab

The Tree Tab displays the call tree data from an executed rule. Each rule occurrence represents a node in the call tree. The tree is arranged hierarchically according to the hierarchy of calls of the corresponding rules. The table consists of the following columns:

Column	Description
Name	Name of the rule/builtin called from this node.
Library	Name of the library from which the rule/builtin named in the **Name** field was invoked. ('*BLTIN*' if RULENAME indicates a builtin).
Type	'Rule' for rule, or 'Routine' for builtin.
Count	Number of times this particular rule/builtin was called from this node.
Total Time	Elapsed time spent in this node and its descendants.
Total Time CPU	Accumulated CPU time spent in this node and its descendants.
Time	Elapsed time spent in this node.
Time CPU	Accumulated CPU time spent in this node.

Totals Tab

The Totals Tab displays a table of the following: A) rules/builtins invoked in the course of an application tracing, or B) a rule from which such a rule has been invoked. There may be 0 or more occurrences of B following each occurrence of A.

The table consists of the following columns:

Column	Description
Name	Name of the rule/builtin called from this node or a rule from which such has rule has been invoked (for a child node).
Library	Name of the library from which the rule/builtin named in the **Name** field was invoked. ('*BLTIN*' if RULENAME indicates a builtin).
Type	'Rule' for rule, or 'Routine' for builtin.
Count	Number of times this particular rule/builtin was called from this node.
Total Time	Elapsed time spent in this node and its descendants.
Total Time CPU	Accumulated CPU time spent in this node and its descendants.

Sorting

You can sort the table by clicking on the header of the column by which to sort. Repeating the click reverses the sort.

Reports View

The Reports view displays a table that lists the reports defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the report.
Unit	Name for a group of related objects.
Modified	Date the report was last modified.
Modifier	Id of the user who last modified the report.
Summary	Explanation of the report.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the reports that appear in the table. For details, see View Filter and Refresh.

Working with Reports

To perform a task on a report, locate the desired report in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired report.
- Right-click the text in the Name column of the desired report.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected report in the TN3270 Report editor.
Add to Project…	Adds the selected report to the named OSB project. This option is valid for multiple selections.

Task Selection	Description
Delete	Deletes the selected report. This option is valid for multiple selections.
Rename	Renames the selected report. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected report into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the report you have copied. When a prompt appears, type a new name for the report you are pasting in the **New name** field.

Rules View

The Rules view displays a table that lists the rules defined in a particular library on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the rule.
Unit	Name for a group of related objects.
Summary	Explanation of the rule.
Modified	Date the rule was last modified.
Modifier	Id of the user who last modified the rule.
Created	Date the rule was created.
Author	Id of the user who created the rule.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the rules that appear in the table. For details, see View Filter and Refresh.

Working with Rules

To perform a task on a rule, locate the desired rule in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired rule.
- Right-click the text in the Name column of the desired rule.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected rule in the Rule editor.
Add to Project…	Adds the selected rule to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected rule. This option is valid for multiple selections.
Rename	Renames the selected rule. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected rule into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the rule you have copied. When a prompt appears, type a new name for the rule you are pasting in the **New name** field.
Show in Totals	Shows a selected rule in the Profiler view Totals tab.
Run...	Run the selected rule using the displayed Run Rule dialog.
Run As	This is a standard Eclipse menu that allows you to run a rule using a specified Run Configuration Type. **OSB Rule** is the applicable rule type.
Debug As	This is a standard Eclipse menu that allows you to run a rule using a specified Debug Configuration Type. **OSB Rule** is the applicable rule type.

Screens View

The Screens view displays a table that lists the screens defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the screen.
Unit	Name for a group of related objects.
Modified	Date the screen was last modified.
Modifier	Id of the user who last modified the screen.
Summary	Explanation of the screen.
Created	Date the screen was created.
Author	Id of the user who created the screen.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the screens that appear in the table. For details, see View Filter and Refresh.

Working with Screens

To perform a task on a screen, locate the desired screen in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired screen.
- Right-click the text in the Name column of the desired screen.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected screen in the TN3270 Screen editor.

Task Selection	Description
Add to Project…	Adds the selected screen to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected screen. This option is valid for multiple selections.
Rename	Renames the selected screen. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected screen into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the screen you have copied. When a prompt appears, type a new name for the screen you are pasting in the **New name** field.

Tables View

The Tables view displays a table that lists the tables defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the table.
Unit	Name for a group of related objects.
Type	Type of the table.
Summary	Explanation of the table.
Created	Date the table was created.
Author	Id of the user who created the table.
Modified	Date the table was last modified.
Modifier	Id of the user who last modified the table.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the tables that appear in the table. For details, see View Filter and Refresh.

Working with Tables

To perform a task on a table, locate the desired table in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired table.
- Right-click the text in the Name column of the desired table.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected table in the table editor corresponding to the table type.
Add to Project…	Adds the selected table to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected table. This option is valid for multiple selections.
Copy table definition	Copies the reference to the definition of the selected table to the system clipboard. This option is valid for multiple selections.
Paste	Pastes the table definition you have copied. Note that **table data will not be copied** into the new table. When a prompt appears, type a new name for the table you are pasting in the **New name** field.
Browse Data	Browse the data in the selected table using the displayed Table Data browser.
Edit Data	Edit the data in the selected table using the displayed Table Data editor

Transactions View

The Transactions view displays a table that lists the transactions defined for an application. The table consists of the following columns:

Columns	Description
Name	Name of the transaction.
Title	Title of the transaction.
Description	Description of the transaction.
Unit	Name for a group of related objects.
Created	Date the transaction was created.
Author	Id of the user who created the transaction.
Modified	Date the transaction was last modified.
Modifier	Id of the user who last modified the transaction.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the transactions that appear in the table. For details, see View Filter and Refresh.

Working with Transactions

To perform a task on a transaction, locate the desired transaction in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired transaction.
- Right-click the text in the Name column of the desired transaction.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected transaction in the Transaction editor.
Add to Project…	Adds the selected transaction to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected transaction. This option is valid for multiple selections.
Rename	Renames the selected transaction. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected transaction to the system clipboard. This option is valid for multiple selections.
Paste	Pastes the transaction you have copied. When a prompt appears, type a new name for the transaction you are pasting in the **New name** field.
Run...	Run the selected transaction using the displayed Run Transaction dialog.
Run As	This is a standard Eclipse menu that allows you to run a transaction using a specified Run Configuration Type. **OSB Transaction** is the type applicable to transactions.

XML Documents View

The XML Documents view displays a table that lists the XML documents defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the XML document.
Unit	Name for a group of related objects.
Type	Type of XML document
Title	Title of the XML document.
Description	Description of the XML document.
Created	Date the XML document was created.
Author	Id of the user who created the XML document.
Modified	Date the XML document was last modified.
Modifier	Id of the user who last modified the XML document.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the XML documents that appear in the table. For details, see View Filter and Refresh.

Working with XML Documents

To perform a task on a XML document, locate the desired XML document in the table, then open the *task menu*. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired XML document.
- Right-click the text in the Name column of the desired XML document.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected XML document in the XML Document editor.
Add to Project…	Adds the selected XML document to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected XML document. This option is valid for multiple selections.
Rename	Renames the selected XML document. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected XML document into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the XML document you have copied. When a prompt appears, type a new name for the XML document you are pasting in the **New name** field.
Run...	Run the selected XML document using the displayed Run XML Document dialog.
Run As	This is a standard Eclipse menu that allows you to run an XML document using a specified Run Configuration Type. **OSB XML document** is the type applicable to XML documents.

XML Field Maps View

The XML Field Maps view displays a table that lists the XML field maps defined on the TIBCO Object Service Broker server. The table consists of the following columns:

Columns	Description
Name	Name of the XML field map.
Unit	Name for a group of related objects.
Title	Title of the XML field map.
Description	Description of the XML field map.
Created	Date the XML field map was created.
Author	Id of the user who created the XML field map.
Modified	Date the XML field map was last modified.
Modifier	Id of the user who last modified the XML field map.

Sorting

You can sort the table by clicking on the header of the column you want to sort the table by. If you click another time, you reverse the sort.

Filtering

You can specify a filter to restrict the XML field maps that appear in the table. For details, see View Filter and Refresh.

Working with XML Field Maps

To perform a task on a XML field map, locate the desired XML field map in the table, then open the *task menu*. You can open the task menu using either of the following methods:

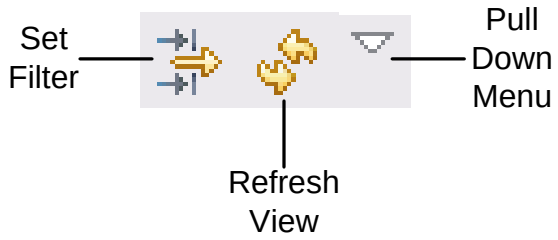
- Click the icon on the left of the Name column of the desired XML field map.
- Right-click the text in the Name column of the desired XML field map.

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected XML field map in the XML Field Map editor.
Add to Project…	Adds the selected XML field map to the named OSB project. This option is valid for multiple selections.
Delete	Deletes the selected XML field map. This option is valid for multiple selections.
Rename	Renames the selected XML field map. When a prompt appears, type a new name in the **New name** field.
Copy	Copies a reference to the selected XML field map into the system clipboard. This option is valid for multiple selections.
Paste	Pastes the XML field map you have copied. When a prompt appears, type a new name for the XML field map you are pasting in the **New name** field.

View Filter and Refresh

The toolbar for views such as the Libraries view or XML Documents view contains the tools shown below:



You use these tools to specify a filter to restrict the objects (such as libraries or XML documents) that appear in a view table.

- **Set TIBCO Object Service Broker Filter icon** – displays the Filter dialog, in which you to specify the filter settings that restrict the objects that appear in a view table.
- **Refresh View icon** – re-fetches a set of objects from the back-end based on the current filter settings.
- **Pull down menu** – the two selections on this menu provide the same functionality as the icons: the **TIBCO Object Service Broker Filter** selection displays the Filter dialog; the **Refresh** selection applies a filter and updates the view table.

Chapter 2 Wizards

This chapter describes the wizards provided by the TIBCO Object Service Broker UI.

New TIBCO Object Service Broker Project Wizard

This wizard creates a new TIBCO Object Service Broker project.

Field or Checkbox	Description
Project name	Name of the new project.
Use default location	Check if you want to use you the default workspace folder you specified when you started Eclipse. You can also specify another workspace folder by typing its path in the **Location** field or by clicking **Browse** to navigate to the folder you want.
Host	Name of where your server resides.
User ID	User ID required to access your server.
Password	Password required to access your server.
Port	Port number required to access your server.
Library	Name of the library associated with the new project.
Debugger Host	Optional host name of this machine. Debugger uses this name to establish a call-back connection from the Execution Environment back to UI. If left blank, the full TCP/IP name is used. See Debugger Connections below for more details.
Debugger Port	A local TCP/IP port that Debugger will use to establish a call-back connection from the Execution Environment back to UI. This field must be specified. See Debugger Connections below for more details.

Import From... Button

Click this button if you already have a TIBCO Object Service Broker project and would like to reuse connection information and other properties applicable to this version of the UI project.

You can see the values corresponding to an existing project by selecting that project in the OSB Project view, then selecting **Project > Properties** on the main menu bar. Select **OSB Properties** in the left pane of the **Properties** dialog.

Test Connection Button

Click this button to test the connection configuration fields. If the test is unsuccessful, check the values you have typed and try again.

TN3270 Editor Settings

You can select different values for **Model**, **Font**, and **Font style**, to adjust the appearance of the TN3270 editors in this project.

To create the new project, click **Finish**, or click **Cancel** to exit.

Debugger Connections

OSB Debugger sessions require additional communication with UI. This requires a special socket connection that the Object Service broker Execution Environment establishes with the UI. As a result of firewall and VPN settings, the client machine may not be visible to the Execution Environment, or the default port may be blocked. In order to avoid these issues you can customize the host and port that Execution will use to connect back to UI.

- **Debugger Host** is an alternative TCP/IP host name. This field is optional. If left blank, the full name of the client machine is used. For example, `myhost.mycompany.com`.
- **Debugger Port** is a port name. It must be specified has a default value of 10001.

New Rule Wizard

This wizard creates a new rule. Rules are the native programming procedures for TIBCO Object Service Broker. Rules tell TIBCO Object Service Broker what tasks to perform, such as what data to access from which database tables, and what to do with the data.

Field	Description
Create in project	Project in which to create the new rule.
Name	Name of the new rule.
Library	Library in which to create the new rule. Click the **Browse** button to select from the list of existing libraries.

To create the new rule, click **Finish**, which displays the Rule editor. Click **Cancel** to exit.

The New Rule wizard will be displayed when you attempt to open a non-existent rule.

Library Selection

Generally, you can create a rule in any accessible library. However, there are some limitations. System and installation libraries play a special role in TIBCO Object Service Broker and you cannot create rules directly in those libraries; therefore, those libraries cannot be specified in New Rule wizard. They are also not available for selection when you click the **Browse...** button.

If you need to create a rule in a special library, you can specify that library in the **Library** field on the OSB Properties page in your project properties. In this case, this library will be accepted as valid. However, this is not a recommended practice and a warning will be displayed.

New Table Wizard

This wizard creates a new table. A table is a logical view of stored data. At a conceptual level, each table is comprised of rows and columns. A row is referred to as an occurrence and a column is referred to as a field. You must create a table definition before the table can be used by other objects.

In TIBCO Object Service Broker, tables are used to store data that is native to TIBCO Object Service Broker, access data that is external to TIBCO Object Service Broker, and access data distributed among multiple TIBCO Object Service Broker systems.

Field	Description
Create in project	Project in which to create the new table.
Name	Name of the new table.
Table Type	Type of table to create. **Note**: For table types DAT, EXP, IMP, MAP, and VSM, you can click **Next** to use Data Discovery. For details, see Chapter 4, Data Discovery.

To create the new table, click **Finish**, which displays the table editor that corresponds to the selected table type. Click **Cancel** to exit.

The New Table wizard will be displayed when you attempt to open a non-existent table.

Table Editors:

DB2 Table Editor
EES Table Editor
EXP Table Editor
IMP Table Editor
MAP Table Editor
PRM Table Editor
SES Table Editor
SUB Table Editor
TDS Table Editor
TEM Table Editor
VSM Table Editor
Other TN3270 Editors

New Transaction Wizard

This wizard creates a new transaction. Transactions are the basic building blocks of an Object Integration Gateway application. They determine what data is accessed, and what actions are performed on the data. All Object Integration Gateway transactions use the following components:

- Interface tables used to supply input and output data.
- Data access specifications (parameter and key) used to select the appropriate data from the database (optional).
- Rules that perform actions on the data.

The wizard creates all the data interfaces and rules required for your transaction.

Field or Checkbox	Description
Create in project	Project in which to create the new transaction.
Name	Name of the new transaction.
Application	Name of the application in which to create the new transaction.
BW Transaction	Specifies if the transaction can be used with TIBCO BusinessWorks™. For details, refer to the TIBCO BusinessWorks documentation that is included in the TIBCO Object Service Broker documentation set.

To create the new transaction, click **Finish**, which displays the Transaction Editor. Click **Cancel** to exit.

The New Transaction wizard will be displayed when you attempt to open a non-existent transaction.

New Application Wizard

This wizard creates a new application. Applications provide a logical way of organizing your transactions. Applications are simply containers for transactions, and every transaction you define must be contained in an application.

Field	Description
Create in project	Project in which to create the new application.
Name	Name of the new application.

To create the new application, click **Finish**, which displays the Application Editor. Click **Cancel** to exit.

The New Application wizard will be displayed when you attempt to open a non-existent application.

New Library Wizard

This wizard creates a new library. Libraries are the basic units of organization for rules, and every rule must be contained in a library.

In the TIBCO Object Service Broker UI, the Libraries view enables you to see the libraries that have been defined, and the rules that each library contains. You can also define new libraries and delete libraries.

Field	Description
Create in project	Project in which to create the new library.
Name	Name of the new library.

To create the new library, click **Finish**, which displays the TN3270 Library editor. Click **Cancel** to exit.

New Object Set Wizard

This wizard creates a new object set. An object set is a collection of TIBCO Object Service Broker objects.

Field	Description
Create in project	Project in which to create the new object set.
Name	Name of the new object set.

To create the new object set, click **Finish**, which displays the TN3270 Object Set editor. Click **Cancel** to exit.

New Report Wizard

This wizard creates a new report. A report is a visual arrangement of data for display in a text-based TIBCO Object Service Broker application. You use the Report editor to define the layout of the report and how its values are calculated

Field	Description
Create in project	Project in which to create the new report.
Name	Name of the new report.

To create the new report, click **Finish**, which displays the TN3270 Report editor. Click **Cancel** to exit.

New Screen Wizard

This wizard creates a new screen. Text-based TIBCO Object Service Broker applications use screens to convey information to and from users. A screen is built from screen tables, which are specialized tables defined through the TN3270 Screen editor.

Field	Description
Create in project	Project in which to create the new screen.
Name	Name of the new screen.

To create the new screen, click **Finish**, which displays the TN3270 Screen editor. Click **Cancel** to exit.

New XML Document Wizard

This wizard creates a new XML document. The Object Integration Gateway can both consume (read) and produce (write) XML documents. When consuming an XML document, the Object Integration Gateway assumes that the data in the document has a relational structure. If the document data does not have a relational structure, or otherwise cannot be successfully mapped to the target data structure, you may need to perform intermediate processing to restructure the data into a relational form, or at least into a form that can be mapped to the target data structure.

Field	Description
Create in project	Project in which to create the new XML document.
Name	Name of the new XML document.

To create the new XML document, click **Finish**, which displays the XML Document Editor. Click **Cancel** to exit.

The New XML Document wizard will be displayed when you attempt to open a non-existent XML Document.

New XML Field Map Wizard

This wizard creates a new XML field map. An XML field map has two complementary purposes. First, it allows the definition of the data to be included as part of the production of an XML document, and to determine how this data is represented in the document. Second, it defines the way that data parsed from an XML document is processed and where the data is placed.

Field	Description
Create in project	Project in which to create the new XML field map.
Name	Name of the new XML field map.

To create the new XML field map, click **Finish**, which displays the XML Field Map Editor. Click **Cancel** to exit.

The New XML Field Map wizard will be displayed when you attempt to open a non-existent XML field map.

Chapter 3 Table Editors

This chapter describes the table editors provided by the TIBCO Object Service Broker UI.

Topics

DAT Table Editor

Use this editor to manage the properties, parameters, fields, and event rules of DAT tables. You can also record the metadata on a DAT table through the Documentation tab.

Table Properties

Type The table type (read-only).

Server ID The ID for the server or group of servers associated with the table. This ID must match the `SERVERID` parameter in the server JCL.

DBID The CA Datacom identification number for the CA Datacom database that is in use. This property is optional.

Dictionary Name The Data Dictionary name of the CA Datacom table to be accessed. A valid entry is any valid CA Datacom table name. This property is optional.

URT Name The name of the URT. Ensure that a load module with this name and the default entry point `DBNTRY` is available to load for the server at runtime. This property is optional.

Internal Name The CA Datacom name that uniquely identifies the table within the CA Datacom database being used. This property is required.

Duplicate Key The option to select if duplicate master keys are allowed for the CA Datacom table. Deselect this option if only a single master key is allowed for that table.

Data Cleansing The data-cleansing actions, if any, to be carried out by the server. This property is optional.

Monitor If monitoring is selected for a table, the current member statistics are compared with the saved statistics for the table. In case of a difference, a warning message is issued. Monitoring is available on z/OS only. This property is not shown if your project is connected to a non-z/OS TIBCO Object Service Broker installation. If a copybook is not selected, the checkbox is disabled.

Element List The CA Datacom element list used by the server at runtime when this table is accessed. This field, which is read-only, is calculated according to the values of the `Element` attribute of fields.

Table Parameters

DAT tables can only have a location parameter.

Location The parameter for accessing external data through a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete this parameter. If you always access the external file remotely, the node from which you request the access might have either of the following definitions:

- **Minimal Definition** — Specify this definition if you always access data from a remote node. This definition consists of the following:
  - — The table name, which must be the same at both locations.
  - — The location parameter, which must also be the same at both locations. You must specify the name of the remote node in which the full definition is located in the Default field, the Source field, or the Source and Sourcename fields.

    The table type in a minimal definition need not match the table type of the full definition on the remote node.
- **Full Definition** — Specify this definition if you can access data at either the local or remote node. The table type of the full definition must match the data on the local node. For example, a full definition of type TDS for accessing TDS data on the local node can also access a DAT table with the same name on a remote node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

A field definition consists of both the TIBCO Object Service Broker and CA Datacom definitions. The TIBCO Object Service Broker field definition is extended by the CA Datacom field definition.

Name The name of the field that must be unique within the table. You can use a name from another table. If you are moving data between this table and another one, assigning the same name to the fields simplifies the process.

Type The TIBCO Object Service Broker UI semantic data type of the field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, see the *TIBCO Object Service Broker Programming in Rules* manual.

The valid values are as follows:

- Blank (Typeless) — The literals defined in TIBCO Object Service Broker rules and fields without a semantic data type.
- **C** (Count) — Integer numbers only (that is, no fractions) for integral count.
- **D** (Date) — The date. You must specify at least the year. The valid dates range from 0001/01/01 to 9999/12/31 inclusive.
- **I** (Identifier) — A unique identifier.
- **L** (Logical) — Information that meets the Yes or No conditions.
- **Q** (Quantity) — Real numbers for measurement.
- **S** (String) — Character string data.

For information on the default mapping of Datacom data types to TIBCO Object Service Broker semantic types and syntax, see Translation of Data Types.

Syntax The TIBCO Object Service Broker syntax of the CA Datacom field. The valid entries are B, C, P, V, F, W, UN, and RD. For information on the default mapping of Datacom data types to TIBCO Object Service Broker semantic types and syntax, see Translation of Data Types.

Length The TIBCO Object Service Broker length (in bytes) of the Datacom field. For information on the default mapping of Datacom data types to TIBCO Object Service Broker semantic types and syntax, see Translation of Data Types.

Decimal The number of digits to the right of the decimal point. This field is relevant only for syntax P. The valid entries are as follows:

- **Syntax P** — This value must be smaller than twice the length of the entire field.
- **Syntax B, C, W, and V** — 0.
- **Semantic type C** — 0.

Key Type The value that uniquely identifies each occurrence in the table. You must define at least one field as a primary key field. The primary key can be a single field or a composite of up to 16 fields, with a maximum length of 127 bytes. There are two choices:

- **P** — The primary key.
- **Blank** — The non-key field.

This column has an associated right-click menu (Move primary keys to the top), which consolidates the primary-key fields at the top of the table. If the primary-key fields are not at the top of the definition, it cannot be saved. This action quickly resolves the error.

Order

The ordering of the field. Select either of the following:

- **A** — Ascending
- **D** — Descending

The default empty value returns occurrences in ascending order by primary key. If you specify ordering for multiple fields, the sort precedence is determined by the order of the fields as listed in the table. Specifying a value in this field incurs sorting overhead, which could be significant in tables with a large number of occurrences.

Required

An indication of whether a value is required for each occurrence in the table. Select True or False or leave this field blank. For primary keys, this field is ignored.

Ext. Syntax, Ext. Length, Ext. Decimal, and Ext. Offset

When reading or writing data from or to CA Datacom, the server converts the data from or to the format described by the following attributes. Exercise caution if you change them.

- **Ext. Syntax** — TIBCO Object Service Broker's external syntax that maps the CA Datacom data type of the field in the CA Datacom table row.
- **Ext. Length** — The length (in bytes) of the field in the CA Datacom table row.
- **Ext. Decimal** — The number of digits to the right of the decimal point in the field in the CA Datacom table row.
- **Ext. Offset** — The offset of the field in the CA Datacom table row. You cannot change this value, which has been computed by either the metadata extractor or the UI Table Editor, depending on which of those two tools you used to generate the definition.

Element

A collection of fields in the CA Datacom rows. Enter the five-character element name, once per element, in the row that corresponds to the field with the lowest row offset (Ext. Offset) within that element. The Element List property displays all the element names you have entered.

Datacom/ Copybook Name

The name that originates from either the CA Data Dictionary or, if you created your DAT definition through the TIBCO Object Service Broker Eclipse UI table definer, from the COBOL copybook you used.

Default

The default value. If you specify a default value for a field but provides no data, the default value (not a null value) is in effect. A valid entry is any valid value for the field. If arithmetic operations are to be performed on numeric fields of type Q or C, you must enter a numeric default value, such as 0.00. Arithmetic operations cannot be performed on data that contains null values.

Reference The name of the table to be referenced when a user is inserting or replacing a value in the field. The added or modified value must exist as a primary key value in the referenced table; otherwise, the action fails.

Reference checking is not done if a null value is specified for a field that is not required and that does not have a default value. A valid entry is the name of any table except a table of type SCR, RPT, or EXP. The primary key field on the table must hold the values for the referenced field. The table cannot be parameterized.

Field Controls The buttons for adding, moving, or deleting fields. For details, see Parameter and Field Actions.

Overlapping Fields

If the Ext. Length or Ext. Offset of some of your fields refers to overlapping areas, updating data with this definition might cause data corruption.

After saving Table Editor will check for overlaps and issue a warning listing all overlapping fields.

In you need have access to this listing after closing the warning dialog box, you can view them in the Error Log view. Choose Window > Show View > Error Log to open that view.

Note that this warning does not prevent saving of the table definition.

Event Rules

To manage the event rules for a DAT table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a DAT table, see Documentation Page.

Translation of Data Types

The following table illustrates CA Datacom data types that can be converted to TIBCO Object Service Broker syntax. Default translations are shown in the following table.

CA Datacom			TIBCO Object Service Broker			
TYPE	SIGN	LENGTH	Xsyn	Syn	Len	Meaning
B	N	2	K	B	3	halfword unsigned
B	N	3	H	V	6	time, HHMMSS
B	N	4	K	B	5	word, unsigned
B	N	8	K	B	9	doubleword, unsigned
B	N	10	H	V	20	timestamp, CCYYMMDDHHMMSSNNNNNN
B	Y	2	B	B	2	halfword, signed
B	Y	4	B	B	4	word, signed
B	Y	8	B	B	8	doubleword, signed
C	N	32719[1]	V	V	L	char
D	N	16[1]	U	P	L	packed, unsigned
D	Y	16[1]	P	P	L	packed, signed
DATE	N	4	H	V	8	date, CCYYMMDD
E	Y	16	F	F	16	expanded float
G	N	32718[1]	UN	UN	L	graphic
H	N	32718[1]	V	V	L	hexadecimal, two-byte display
K	N	32718[1]	UN	UN	L	Kanji, same as G
L	Y	8	F	F	8	long float
N	N	31[1]	M	P	L/2+1	zoned decimal, unsigned
N	Y	31[1]	N	P	L/2+1	zoned decimal, signed
RAW	N	32719[1]	H	RD	L+4	all "other" data types
S	Y	4	F	F	4	small float
Y	N	32718[1]	UN	UN	L	DBCS, same as G
Z	N	32719[1]	W	W	L	mixed (DBCS & SBCS)
2	N	2	K	B	3	halfword, unsigned
2	Y	2	B	B	2	halfword, signed
4	N	4	K	B	5	word, unsigned
4	Y	4	B	B	4	word, signed
8	N	8	K	B	9	doubleword, unsigned
8	Y	8	B	B	8	doubleword, signed

[1]The reported/assumed length of the CA Datacom field is less or equal to this number. The notation L in the Len column refers to this length.

DB2 Table Editor

This editor is used to define properties, parameters, fields, and event rules of TIBCO Object Service Broker tables, mapping DB2 tables and stored procedures. You can also record metadata about a DB2 table using the Documentation tab.

See Also *TIBCO Service Gateway for DB2 Installing and Operating* for details on defining DB2 tables.

Table Properties

These properties are searchable; you can search for an application based on one property or a combination of these properties.

Type Table type text (read-only, always 'DB2').

Subtype DB2 table subtype drop down. There are three different subtypes of DB2 table types: **N – Table**; **S – Procedure**; and **R – Result Set**. The Result Set value is not selectable and cannot be changed.

Note: The **DB2 Creator / Schema** and **Name** fields will be cleared and all selected fields in the Fields section will be removed after the table subtype is changed. Selecting the Procedure subtype will also enable stored procedure specific controls and fields.

Dictionary Specifies whether to use the Internal (I) or External (X) dictionary.

Server Type DB2 server type with the maximum length of 3 characters (read-only). Valid values: **DB2** or **IM2**.

Server ID DB2 server ID with the maximum length of 8 characters.

DB2 Creator / Schema DB2 creator or schema (read-only).

Implied Update Indicates whether TIBCO Object Service Broker is expected to consider the fact of invocation of this stored procedure as an update operation, meaning the transaction in whose context it occurred should be reported to the Data Object Broker as resulting in data changes; this is essential for determining the specifics of commit/rollback/recovery procedures to be used for this transaction. Note that not selecting this attribute improves the application's performance (unless other modifications of data, either implicit or explicit, occur within the boundaries of the same transaction).

DB2 Location DB2 server location with the maximum length of 16 characters.

Table / Procedure DB2 table, stored procedure or result set name (read-only). Use the Search button to load a valid table or stored procedure name.

Note: All selected fields in the Fields section will be removed after the DB2 table or stored procedure name is changed. Fields of the stored procedure table will be also pre-populated.

Orders When checked, TIBCO Object Service Broker orders the table data.

Search Opens a dialog to select a DB2 table or stored procedure linked with this table.

Stored Procedure-specific Controls and Fields

Result Set List Opens a dialog to view and select TIBCO Object Service Broker DB2 Result Set tables associated with this table in the Result Set Selector window.

Refresh Result Sets Automatically creates new result sets table definitions for the stored procedure. The Stored Procedure Parameters dialog window will be displayed, if needed, to specify the stored procedure parameters.

Result set table names will be constructed from the first 11 characters of the stored procedure table name following a "_CSR" and a sequence number. For example: a table called SPTEST will have result set tables call SPTEST_CSR1, SPTEST_CSR2, etc. for the number of result sets.

Note: Make the first 11 characters of your stored procedure table name unique to avoid name conflicts for the generated result sets.

Warning: Upon the first invocation of creating result set definitions the stored procedure must be executed. This is required in order to describe the result sets. Be aware that by executing the stored procedure you may update data. If this is not acceptable you may consider defining the result set manually.

#Parameters The number of stored procedure parameters (read-only).

#ResultSets The number of result sets for this stored procedure (read-only).

CommitOn Return The logical value stored in DB2's table SYSROUTINES.

Note: All attributes of the result set tables specified in the Properties section are non editable and disabled. Also disabled are buttons in the Parameters and Fields sections. The only editable area of a result set table is the metadata columns, which are defined in the same way as for other table types.

Result Set-specific Controls and Fields

Cursor Name The procedure-assigned name cursor name.

Table Parameters

You can define the Location parameter only:

Location Used to access external data through a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete the parameter. If you always access the external file remotely, the node from which you request the access may have either a minimal or full definition.

- Minimal Definition – used when you always access data at a remote node. It consists of the following:
  - The table name, which must be the same at both locations.
  - The location parameter, which must be the same at both locations. The name of the remote node where the full definition is located must be supplied in the Default field, Source field, or Source and Sourcename fields.

  The table type specified in a minimal definition need not match the table type of the full definition on the remote node.
- Full Definition – used when you can access data at either the local or the remote node. The table type of the full definition must match the data on the local node. For example, a full definition of type TDS used to access TDS data on the local node may also be used to access a DB2 table with the same name on a remote node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a DB2 table, the following apply:

- The DB2 table fields consist of two groups of columns: DB2 table columns (on the left) and metadata columns (on the right). You cannot change any of the information in the DB2 table columns. Data in the metadata columns are defined in the same way as for other table types.
- When defining fields, you can select external attributes (External Syntax, External Length and External Decimal) and the UI will select reasonable defaults for internal attributes (Syntax, Length and Decimal) and vice versa. If these values are not satisfactory, they can be re-selected manually.

- For a DB2 stored procedure table, the columns that are displayed are the parameters of the stored procedure.
- Two additional TIBCO Object Service Broker fields are added to the definition of a DB2 stored procedure table: @HANDLE@ and @RESULT@.
- All DB2 stored procedure table fields are pre-selected. No fields can be removed and no new fields can be added to ensure that a call to the stored procedure does not fail.

Selection The way of using the DB2 field in the TIBCO Object Service Broker table. Valid values: **S – Select**; **K – Key**; **P – Parameter**.

DB2 Name The DB2 table field name (read-only).

Datatype Type of the DB2 field (read-only).

DB2 Length Length of the DB2 field (read-only).

DB2 Scale Scale value of the DB2 field (read-only).

DB2 Default DB2 default value for the field, if no data is available.

Name Name of the field that must be unique within the table. The maximum length of a TIBCO Object Service Broker field name is 16.

Type The TIBCO Object Service Broker UI semantic data type of the field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax The TIBCO Object Service Broker UI syntax of the field. You can specify any valid semantic type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*.

Length Length of the DB2 field. The data is padded or truncated as necessary.

Decimal Number of digits to appear to the right of the decimal point. The data is padded or truncated as necessary.

Order The order in which the occurrences in this field are sorted. Select **A – Ascending** or **D – Descending** or leave blank.

Required Specifies if a value is required for each occurrence in the table. Select true or false or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a DB2 table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a DB2 table, see Documentation Page.

EES Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of EES tables. You can also record metadata about a EES table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked).

Table Parameters

You can define two types of table parameters:

Data You can specify a maximum of four data parameters for a table, to a total maximum length of 240 bytes.

Location Must be the last parameter name listed. For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining an EES table, the following apply:

- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE.
- EES tables have two special fields always present at the end of the list of fields: @@UPDATE_COUNT and @@REF_COUNT. These fields cannot be deleted or modified in any way and no fields can be added after them.

These descriptions define the attributes of both primary key and non-key fields; differences between primary and non-key fields are noted below.

Name Name of field that must be unique within the table.

Type Semantic data type of the field. For primary keys, if the table has the IDgen property selected, must be **I – Identifier**. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for

integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of the field. For primary keys, if the table has the IDgen property selected, must be **B** – binary.

Length Length of the field. The value is in bytes and valid values are determined by the syntax of the field. For valid lengths, refer to *TIBCO Object Service Broker Programming in Rules*. If the table has the IDgen option selected, the length must be 4 bytes.

Decimal Number of digits to appear to the right of the decimal point. In most cases this is optional. It is only relevant for Syntax **P** – packed decimal.

Key Type Specifies the primary key. At least one primary key field must be specified. You can specify a single field or up to sixteen fields to be the primary key, to a total maximum length of 127 bytes. The primary key is always stored as the first field or series of fields.

If the table has the IDgen property selected, the primary key field must be the first field in the definition, must have type I, syntax B and length 4, and one and only one key field must be specified.

Order The ordering of the field. Select **A – Ascending** or **D – Descending**. The default empty value returns occurrences in ascending order by primary key. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table. Specifying a value in this field incurs sorting overhead, which may be significant in tables with a large number of occurrences.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for an EES table, see Table Editor Event Rules.

Documentation

To manage the description and usage for an EES table, see Documentation Page.

EXP Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of EXP tables. You can also record metadata about an EXP table using the Documentation tab.

Table Properties

Type Table type (read-only).

DDName Name that points to the data set or file to which you are exporting the data. The way you associate the DDname value with a file name depends on the platform the TIBCO Object Service Broker UI system to which you are connected is running.

On z/OS, enter the name of a DD statement in the JCL used to start the Execution Environment.

On Windows, Solaris, and Linux, use one of the following:

- Enter the name of an environment variable that is set to the fully qualified name of the file. The environment variable must be available to the Execution Environment at startup.
- Enter the name of a DD definition in the huron.dsn file.

If you specify a DDname value, the export table cannot have parameters. If you are writing multiple export tables to a single file, the DDname value in all the export tables must be the same. If you specify both a File name and a DDname, the File name is used.

External routine name Name of an external routine that is a load module resulting from an assembler program. The routine resides in a data set that is concatenated to the external utilities data set. You can use an external routine to manipulate data before exporting it to the destination data set. Valid only if the TIBCO Object Service Broker UI system you are connected to is running on the z/OS platform.

Server ID If the table is to be accessed remotely via the TIBCO Service Gateway for Files, specify the server ID to be used; otherwise, leave blank.

Monitor If you select monitoring for a table, the current member statistics are compared with the saved statistics for the table. In case of a difference, a warning message is issued. Monitoring is available on z/OS only. This property is not shown if your project is connected to a non-z/OS TIBCO Object Service Broker installation.

If a copybook is not selected, the checkbox is disabled.

File name Name of the data set or file to which you are exporting the data. The maximum record size is 31,744 bytes for all platforms. The record length for fixed length files must be at least equal to the total length of the fields. The record length for variable length files must be at least four more than the total length of the fields. If you specify both a File name and a DDname, the File name is used.

To specify the format of the data in external files to be processed by TIBCO Object Service Broker UI on non-z/OS systems, use the DSIXFTYPE Execution Environment parameter. For more information, see *TIBCO Object Service Broker Parameters*.

If you use FTP to transfer a variable length export table file between z/OS and Windows, Solaris, or Linux, you must reformat the file using the TIBCO Object Service Broker UI z/OS utility HRNBRFRU. If the export file is fixed length, you do not need to reformat the file.

Under CICS, you do not have to define the data set in a Destination Control Table (DCT) but the CICS Region must have external security access to the data set.

This file name maps to an entry in the huron.dsn file on non-z/OS systems. This file describes the absolute path through which this file name can be accessed along with other file attributes. If the file name is mapped to an entry in the file, then any special characters (*, ?, <, >) included in the file name are ignored. If a default path name is used, any special characters in the file name are replaced and you do not receive the intended file name. For additional information on the huron.dsn file, refer to *TIBCO Service Gateway for Files*.

For parameterized export tables, you must specify a partitioned data set or a directory. Neither of these need to exist before you define the export table but must exist before you insert data. Parameter values you provide become the data set member names or the file names in the directory. You can only create one table instance per TIBCO Object Service Broker UI transaction. The parameter must have syntax C. On z/OS, the parameter's length must be 8. In Windows, Solaris, and Linux, its length must be 8 or less.

With non-parameterized export tables you can export data to any file or member of a partitioned data set.

Table Parameters

You can define two types of table parameters:

Data Used if you want to write data to a partitioned data set or a directory. The parameter value you provide becomes the data set member name or the file name in the directory. Data parameters must be defined with syntax C. On z/OS, the parameter length must be 8. In Windows, Solaris, and Linux, the length must be 8 or less.

Location Used to write data to a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete the parameter. If you always access the external file remotely, the node from which you request the access may have either a minimal or full definition.

- Minimal Definition – used when you always access data at a remote node. It consists of the following:
  - The table name, which must be the same at both locations.
  - The location parameter, which must be the same at both locations. The location parameter indicates that you always access data at a remote node. The name of the remote node where the full definition is located must be supplied in the Default field, Source field, or Source and Sourcename fields.

  The table type specified in a minimal definition need not match the table type of the full definition on the remote node.
- Full Definition – used when you can access data at either the local or the remote node. The table type of the full definition must match the data on the local node. For example, a full definition of type TDS used to access TDS data on the local node may also be used to access an export table with the same name on a remote node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining an export table, the following apply:

- When defining fields, you can select external attributes (External Syntax, External Length and External Decimal) and UI will select reasonable defaults for internal attributes (Syntax, Length and Decimal) and vice versa. If these values are not satisfactory they can be re-selected manually.
- When you add a new field UI will generate a default value for Offset, based on the size and offset of preceding fields. If this value in not satisfactory it can be adjusted manually.
- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE.
- Areas of the record not defined as fields contain hex zeros when written.
- If LRECL is not specified in the DCB information at run time, the position of the last byte of the right-most field (as specified in the external field definition) is used to calculate the LRECL.

These fields define the external attributes and internal TIBCO Object Service Broker attributes for the primary key fields and data fields of the table.

If a copybook was selected using the Data Discovery wizard, fields may be selected directly from the data set member or file containing the copybook. The data set or file will be read each time Add from COBOL copybook... is selected. TIBCO Object Service Broker field names, external syntax, external length, external decimal and offset values will be generated from the copybook values.

Name Name of the field that must be unique within the table definition. You can use the same name as a field in any other table. If you are moving data between this table and another table, giving fields the same names simplifies the process.

External Syntax External syntax for the export field. On Windows, Solaris, and Linux, numeric external syntaxes are treated as C or V for "LINE_SEPARATED_ASCII" text files. The external syntaxes E and K are not valid for TIBCO Object Service Broker UI for z/OS.

External Length External length for the export field. On the Windows, Solaris, and Linux platforms, this field is ignored for "LINE_SEPARATED_ASCII" text files with a field separator character defined.

External Decimal External number of decimal places for the export field.

Offset The offset of the export field relative to the start of the external record. The origin is zero. You do not need to define fillers, since the offset can be used to skip undefined locations in the row. On the Windows, Solaris, and Linux platforms, this field is ignored for "LINE_SEPARATED_ASCII" text files with a field separator character defined).

This column has an associated right-click menu (Recalculate offsets), which recalculates all the offsets. All the fields are then positioned sequentially without gaps between them.

Key Type Specifies whether the export fields are to be used in the primary key. You can select any field as the primary key, without respect to uniqueness of data. You can select up to 16 contiguous fields for a composite primary key, to a total maximum length of 127 bytes.

Type The TIBCO Object Service Broker UI semantic data type of the export field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined

without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax The TIBCO Object Service Broker UI syntax of the export field. You can specify any valid semantic type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*.

Length Length of the export field. The data is padded or truncated as necessary.

Decimal Number of digits to be displayed to the right of the decimal point. The data is padded or truncated as necessary.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank.

Default Default value for the field when it is displayed. If no data is available, the value provided in this field is used. For example, if you specify a dot (.) as the default, it is displayed for a field that does not have any values. If you do not specify anything, a blank space is displayed. For numeric fields of Q or C, specify a value such as 0.00 if arithmetic operations are to be performed on the field; arithmetic operations cannot be performed on data containing null values.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for an EXP table, see Table Editor Event Rules.

Documentation

To manage the description and usage for an EXP table, see Documentation Page.

IMP Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of IMP tables. You can also record metadata about an IMP table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked). Required if you will be processing multiple record formats or multiple occurrences of data within a record.

Server ID If the table is to be accessed remotely via the TIBCO Service Gateway for Files, specify the server ID to be used; otherwise, leave blank.

File name Name of the data set or file that contains the data you want to import. The maximum record size is 31,744 bytes for all platforms. The record length for fixed length files must be at least equal to the total length of the fields. The record length for variable length files must be at least four more than the total length of the fields.

To specify the format of the data in external files to be processed by TIBCO Object Service Broker UI on non-z/OS systems, use the DSIXFTYPE Execution Environment parameter. For more information, see *TIBCO Object Service Broker Parameters*.

If you use FTP to transfer a variable length import table file between z/OS and Windows, Solaris, or Linux, you must reformat the file using the TIBCO Object Service Broker UI z/OS utility HRNBRFRU. If the import file is fixed length, you do not need to reformat the file.

This file name maps to an entry in the huron.dsn file on non-z/OS systems. This file describes the absolute path through which this file name can be accessed. For additional information on the huron.dsn file, refer to *TIBCO Service Gateway for Files*.

If you use FTP to transfer from z/OS to Windows, Solaris, or Linux, the transfer must be in binary mode with the z/OS FTP LOCSITE subcommand with the NORDW parameter specified.

For parameterized import tables (except for those used to define multiple record formats), you must specify a partitioned data set or a directory. Neither of these need to exist before you define the import table but must exist before you access

the data. Parameter values you provide become the data set member names or the file names in the directory.

For non-parameterized import tables, you can import data from any file or member of a partitioned data set.

Under CICS, you do not have to define the data set in a Destination Control Table (DCT) but the CICS region must have external security access to the data set.

If you are accessing multiple TIBCO Object Service Broker UI import tables from a single data set, the File name in all the tables must be the same.

If you specify both a File name and a DDname, the File name is used.

DDName Name that points to the data set or file containing the data to be imported by this table. The way you associate the DDname value with a file name depends on the platform the TIBCO Object Service Broker UI system to which you are connected is running.

On z/OS, enter the name of a DD statement in the JCL used to start the Execution Environment.

On Windows, Solaris, and Linux, use one of the following:

- Enter the name of an environment variable that is set to the fully qualified name of the file. The environment variable must be available to the Execution Environment at startup.
- Enter the name of a DD definition in the huron.dsn file.

If you specify a DDname value, the import table cannot have parameters, except for those used to define multiple record formats. If you are accessing multiple import tables from a single file, the DDname value in all the import tables must be the same. If you specify both a File name and a DDname, the File name is used.

External routine name Name of an external routine that is a load module resulting from an assembler program. The routine resides in a data set that is concatenated to the external utilities data set. You can use an external routine to manipulate data in the source file before importing it into TIBCO Object Service Broker UI. Valid only if the TIBCO Object Service Broker UI system you are connected to is running on the z/OS platform.

Data cleansing The data-cleansing actions, if any, to be carried out by the Execution Environment. This property is optional.

Monitor If you select monitoring for a table, the current member statistics are compared with the saved statistics for the table. In case of a difference, a warning message is issued. Monitoring is available on z/OS only. This property is not shown if your project is connected to a non-z/OS TIBCO Object Service Broker installation.

If you do not select a copybook, the checkbox is inactive.

Table Parameters

You can define two types of table parameters:

Data Used if you want to read data from a partitioned data set or a directory. The parameter value you provide becomes the data set member name or the file name in the directory. A data parameter is validated as follows:

- **If Syntax is C and Length is 8** – TIBCO Object Service Broker UI interprets the parameter value as a member name of a partitioned data set (z/OS) or a file within a directory (Windows, Solaris, and Linux).
- **If Syntax is B and Length is 4** – TIBCO Object Service Broker UI interprets the parameter value as an indicator for multiple record types or repeating groups.

If IDgen is selected, you cannot use the data parameter to view different members of a partitioned data set (PDS). If IDgen is not selected, the data parameter must have a Syntax of C and be no longer than 8 characters.

Location Used to access external data through a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete the parameter. If you always access the external file remotely, the node from which you request the access may have either a minimal or full definition.

- Minimal Definition – used when you always access data at a remote node. It consists of the following:
  - — The table name, which must be the same at both locations.
  - — The location parameter, which must be the same at both locations. The location parameter indicates that you always access data at a remote node. The name of the remote node where the full definition is located must be supplied in the Default field, Source field, or Source and Sourcename fields.

  The table type specified in a minimal definition need not match the table type of the full definition on the remote node.
- Full Definition – used when you can access data at either the local or the remote node. The table type of the full definition must match the data on the local node. For example, a full definition of type TDS used to access TDS data on the local node may also be used to access an import table with the same name on a remote node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining an import table, the following apply:

- When defining fields, you can select external attributes (External Syntax, External Length and External Decimal) and UI will select reasonable defaults for internal attributes (Syntax, Length and Decimal) and vice versa. If these values are not satisfactory they can be re-selected manually.
- When you add a new field UI will generate a default value for Offset, based on the size and offset of preceding fields. If this value in not satisfactory it can be adjusted manually.
- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE.

These fields define the external attributes and internal TIBCO Object Service Broker attributes for the primary key fields and data fields of the table.

If a copybook was selected using the Data Discovery wizard, fields may be selected directly from the data set member or file containing the copybook. The data set or file will be read each time Add from COBOL copybook... is selected. TIBCO Object Service Broker field names, external syntax, external length, external decimal and offset values will be generated from the copybook values.

Name Name of the field that must be a unique within the table. You can use a name from another table; if you are moving data between this table and another table, giving fields the same name simplifies the process.

External Syntax External syntax for the import field. On Windows, Solaris, and Linux, numeric external syntaxes are treated as C or V for LINE_SEPARATED_ASCII text files. External syntax **T** is not supported for import tables on z/OS. The external syntaxes E and K are not valid for TIBCO Object Service Broker UI for z/OS.

External Length External length for the import field. On the Windows, Solaris, and Linux platforms, this field is ignored for LINE_SEPARATED_ASCII text files with a field separator character defined.

External Decimal External number of decimal places for the import field.

Offset The offset maps to the start of the external record in the import file. The origin is zero. Overlaps are allowed and you do not need to define fillers, since the offset can be used to skip undefined locations in the occurrence. On the Windows, Solaris, and Linux platforms, the Offset field is ignored for

LINE_SEPARATED_ASCII text files with a field separator character defined.

This column has an associated right-click menu (Recalculate offsets), which recalculates all the offsets. All the fields are then positioned sequentially without gaps between them.

Key Type Specifies whether the import fields are to be used in the primary key. You can select any field as the primary key, without respect to uniqueness of data. You can select up to 16 contiguous fields for a composite primary key, to a total maximum length of 127 bytes.

Type The TIBCO Object Service Broker UI semantic data type of the import field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax The TIBCO Object Service Broker UI syntax of the import field. You can specify any valid semantic type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*.

Length Length of the import field. The data is padded or truncated as necessary.

Decimal Number of digits to appear to the right of the decimal point. The data is padded or truncated as necessary.

Order The order in which the occurrences in this field are sorted. The default empty value returns occurrences in ascending order by primary key. When an ordering option is explicitly specified, it takes precedence over the default. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table.

Specifying a value incurs sorting overhead, which may be significant in tables with a large number of occurrences. On Windows, Solaris, and Linux, if you have an import table with duplicate records and you want to sort, the sort results can be unpredictable since the internal sort does not maintain the input order of duplicate records. You have to specify the ordering for each field to ensure you get the ordering you want. Select **A – Ascending** or **D – Descending** or leave blank.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for an IMP table, see Table Editor Event Rules.

Documentation

To manage the description and usage for an IMP table, see Documentation Page.

MAP Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of MAP tables. You can also record metadata about a MAP table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked). Required for MAP tables.

Data cleansing The data-cleansing actions, if any, to be carried out by the Execution Environment. This property is optional.

Monitor If you select monitoring for a table, the current member statistics are compared with the saved statistics for the table. In case of a difference, a warning message is issued. Monitoring is available on z/OS only. This property is not shown if your project is connected to a non-z/OS TIBCO Object Service Broker installation.

If a copybook is not selected, the checkbox is disabled.

Table Parameters

You can define three types of table parameters:

Address Used to access storage data; must have Syntax of B, a Length of 4, and Class of A.

Count An optional parameter used to limit the number of occurrences eligible for selection by a GET or FORALL statement. This does not mean the number of occurrences that are retrieved; the number of occurrences selected is equal to or lower than the count value, and may be zero. Must have Syntax of B, a Length of 4, and Class of C. If not specified, a default value of infinity is used.

Location Used to access external data through a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete the parameter. If you always access the external file remotely, the node from which you request the access may have either a minimal or full definition. If you use a MAP table remotely, all storage references must be to addresses valid on the remote system

- Minimal Definition – used when you always access data at a remote node. It consists of the following:
  - — The table name, which must be the same at both locations
  - — The location parameter, which must be the same at both locations. The name of the remote node where the full definition is located must be supplied in the Default field, Source field, or Source and Sourcename fields.

  The table type specified in a minimal definition need not match the table type of the full definition on the remote node.
- Full Definition – used when you can access data at either the local or the remote node. The table type of the full definition must match the data on the local node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a MAP table, the following apply:

- When defining fields, you can select external attributes (External Syntax, External Length and External Decimal) and UI will select reasonable defaults for internal attributes (Syntax, Length and Decimal) and vice versa. If these values are not satisfactory they can me re-selected manually.
- When you add a new field UI will generate a default value for Offset, based on the size and offset of preceding fields. If this value in not satisfactory it can be adjusted manually.
- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE. The ESTIMATETBLDFN tool allows you to estimate the size of this parameter.

These fields define the external attributes and internal TIBCO Object Service Broker attributes for the primary key fields and data fields of the table.

If a copybook was selected using the Data Discovery wizard, fields may be selected directly from the data set member or file containing the copybook. The data set or file will be read each time Add from COBOL copybook... is selected. TIBCO Object Service Broker field names, external syntax, external length, external decimal and offset values will be generated from the copybook values.

Name Name of the primary key or data field you are creating. This name must be unique within the table. You can use a name already used for as a field in any other table; if you are moving data between this table and another table, giving fields the same name simplifies the process.

External Syntax External syntax for the field. The external syntaxes **E** and **R** are not valid for TIBCO Object Service Broker UI for z/OS.

External Length External length for the field.

External Decimal External number of decimal places for the field.

Offset External offset of the field based on the length of the field. Overlaps of fields are allowed. You need not define fillers because you can skip undefined locations in the row with the offset. The key field does not participate in offset calculations; the first non-key field has offset 0 by default.

This column has an associated right-click menu (Recalculate offsets), which recalculates all the offsets. All the fields are then positioned sequentially without gaps between them.

Key Type Specifies whether the MAP field is to be used as a primary key. This field must be the first one in the definition and must have type of I, syntax of B and length of 4. Only a single key field can be specified.

Type The TIBCO Object Service Broker UI semantic data type of the field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax The TIBCO Object Service Broker UI syntax of the field. You can specify any valid semantic type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*.

Length Length of the field. The data is padded or truncated as necessary.

Decimal Number of digits to appear to the right of the decimal point. The default is 0. The data is padded or truncated as necessary.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a MAP table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a MAP table, see Documentation Page.

PRM Table Editor

This editor is used to manage the properties of PRM tables. You can also record metadata about a PRM table using the Documentation tab.

Table Properties

Type Table type (read-only)

Source Name of the source table.

Documentation

To manage the description and usage for a PRM table, see Documentation Page.

SES Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of SES tables. You can also record metadata about a SES table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked).

Table Parameters

You can define two types of table parameters:

Data You can specify a maximum of four data parameters for a table, to a total maximum length of 240 bytes.

Location Must be the last parameter name listed.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a SES table, the following applies

- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE.

These descriptions define the attributes of both primary key and non-key fields; differences between primary and non-key fields are noted.

Name Name of field that must be unique within the table.

Type Semantic data type of the field. For primary keys, if the table has the IDgen property selected, must be **I – Identifier**. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L –**

Logical: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of the field. For primary keys, if the table has the IDgen property selected, must be **B** – binary.

Length Length of the field. The value is in bytes and valid values are determined by the syntax of the field. For valid lengths, refer to *TIBCO Object Service Broker Programming in Rules*. If the table has the IDgen option selected, the length must be 4 bytes.

Decimal Number of digits to appear to the right of the decimal point. In most cases this is optional. It is only relevant for Syntax **P** – packed decimal.

Key Type Specifies the primary key. At least one primary key field must be specified. You can specify a single field or up to sixteen fields to be the primary key, to a total maximum length of 127 bytes. The primary key is always stored as the first field or series of fields.

If the table has the IDgen property selected, the primary key field must be the first field in the definition, must have type I, syntax B and length 4, and one and only one key field must be specified.

Order The ordering of the field. Select **A – Ascending** or **D – Descending**. The default empty value returns occurrences in ascending order by primary key. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table. Specifying a value in this field incurs sorting overhead, which may be significant in tables with a large number of occurrences.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a SES table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a SES table, see Documentation Page.

SUB Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of SUB tables. You can also record metadata about a SUB table using the Documentation tab.

Table Properties

Type Table type (read-only)

Source Name of the source table

Select Selection criteria used to select data from the source table

Lock Mode Specifies whether locks are taken.

Table Parameters

You can define two types of table parameters:

Data You can specify a maximum of four data parameters for a table, to a total maximum length of 240 bytes. In addition, the following apply:

- A data parameter of a source table, if it is not defined in the subview, must be selected in the selection area of the subview screen as described above.
- A data parameter in a subview can be a field of the source table, but it must be selected in the selection area of the subview screen as described above.
- The names of the parameters may differ between the subview and the source table. If the names differ, the value in the Source field must be set to 'Source' and the name of the parameter in the source table must be provided in the Source Name field.

Location Must be the last parameter name listed.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a SUB table, the following applies:

- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE. The ESTIMATETBLDFN tool allows you to estimate the size of this parameter.

These descriptions define the attributes of both primary key and non-key fields; differences between primary and non-key fields are noted below. Note the following:

- Key fields in the source table must be defined as key fields in the subview.
- Some or all of the non-key fields named in the subview may be the same as those in the source table.
- Some of the fields in the subview may be renamed from the source table but derive their values from the source table.
- Only the name and primary key setting of the source fields are imported from the source table. The other attributes, such as syntax and length, are left unspecified. Unless you override these other attributes in the subview table, they are inherited from the source table.
- Some of the fields may be new fields unique to the subview and derive their value through a functional rule.

Name Name of field that must be unique within the table.

Type Semantic data type of the field. For primary keys, if the table has the IDgen property selected, must be **I – Identifier**. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of the field. For primary keys, if the table has the IDgen property selected, must be **B** – binary.

Length Length of the field. The value is in bytes and valid values are determined by the syntax of the field. For valid lengths, refer to *TIBCO Object Service Broker Programming in Rules*. For primary keys, must have a specified value, and if the table has the IDgen option selected, the length must be 4 bytes.

Decimal Number of digits to appear to the right of the decimal point. In most cases this is optional. It is only relevant for Syntax **P** – packed decimal.

Key Type Specifies the primary key.

Order The ordering of the field. Select **A – Ascending** or **D – Descending**. The default empty value returns occurrences in ascending order by primary key. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table. Specifying a value in this field incurs sorting overhead, which may be significant in tables with a large number of occurrences.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Documentation

To manage the description and usage for a SUB table, see Documentation Page.

TDS Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of TDS tables. You can also record metadata about a TDS table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked).

Table Parameters

You can define two types of table parameters:

Data You can specify a maximum of four data parameters for a table, to a total maximum length of 240 bytes.

Location Must be the last parameter name listed.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a TDS table, the following applies:

- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE. The ESTIMATETBLDFN tool allows you to estimate the size of this parameter.

These descriptions define the attributes of both primary key and non-key fields; differences between primary and non-key fields are noted below.

Name Name of field that must be unique within the table.

Type Semantic data type of the field. For primary keys, if the table has the IDgen property selected, must be **I – Identifier**. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range

from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of the field. For primary keys, if the table has the IDgen property selected, must be **B** – binary.

Length Length of the field. The value is in bytes and valid values are determined by the syntax of the field. For valid lengths, refer to *TIBCO Object Service Broker Programming in Rules*. If the table has the IDgen option selected, the length must be 4 bytes.

Decimal Number of digits to appear to the right of the decimal point. In most cases this is optional. It is only relevant for Syntax **P** – packed decimal.

Key Type Specifies the primary key. At least one primary key field must be specified. You can specify a single field or up to sixteen fields to be the primary key, to a total maximum length of 127 bytes. The primary key is always stored as the first field or series of fields.

If the table has the IDgen property selected, the primary key field must be the first field in the definition, must have type I, syntax B and length 4, and one and only one key field must be specified.

Order The ordering of the field. Select **A – Ascending** or **D – Descending**. The default empty value returns occurrences in ascending order by primary key. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table. Specifying a value in this field incurs sorting overhead, which may be significant in tables with a large number of occurrences.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a TDS table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a TDS table, see Documentation Page.

TEM Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of TEM tables. You can also record metadata about a TEM table using the Documentation tab.

Table Properties

Type Table type (read-only)

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked).

Table Parameters

You can define two types of table parameters:

Data You can specify a maximum of four data parameters for a table, to a total maximum length of 240 bytes.

Location Must be the last parameter name listed.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a TEM table, the following applies:

- The number of fields you can access is dependent upon the Data Object Broker parameter CTABLESIZE.

These descriptions define the attributes of both primary key and non-key fields; differences between primary and non-key fields are noted.

Name Name of field that must be unique within the table.

Type Semantic data type of the field. For primary keys, if the table has the IDgen property selected, must be **I – Identifier**. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L –**

Logical: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of the field. For primary keys, if the table has the IDgen property selected, must be **B** – binary.

Length Length of the field. The value is in bytes and valid values are determined by the syntax of the field. For valid lengths, refer to *TIBCO Object Service Broker Programming in Rules*. If the table has the IDgen option selected, the length must be 4 bytes.

Decimal Number of digits to appear to the right of the decimal point. In most cases this is optional. It is only relevant for Syntax **P** – packed decimal.

Key Type Specifies the primary key. At least one primary key field must be specified. You can specify a single field or up to sixteen fields to be the primary key, to a total maximum length of 127 bytes. The primary key is always stored as the first field or series of fields.

If the table has the IDgen property selected, the primary key field must be the first field in the definition, must have type I, syntax B and length 4, and one and only one key field must be specified.

Order The ordering of the field. Select **A – Ascending** or **D – Descending**. The default empty value returns occurrences in ascending order by primary key. When ordering is specified for more than one field, the sort precedence is determined by the order of the fields as they are listed in the table. Specifying a value in this field incurs sorting overhead, which may be significant in tables with a large number of occurrences.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank. For primary keys, this field is ignored.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a TEM table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a TEM table, see Documentation Page.

VSM Table Editor

This editor is used to manage the properties, parameters, fields, and event rules of VSM tables. You can also record metadata about a VSM table using the Documentation tab.

Table Properties

Type Table type (read-only).

IDgen TIBCO Object Service Broker generates the value of the primary key field (when checked). Required if you will be processing multiple record formats or multiple occurrences of data within a record.

Read Only Prohibit changes to the table data. If selected, any attempt to change the table data results in an error. Field overlaps are allowed only for read-only tables. If your definition has fields that overlap when the Read Only checkbox is cleared, you are not allowed to save the definition.

Monitor If you select monitoring for a table, the current member statistics are compared with the saved statistics for the table. In case of a difference, a warning message is issued. Monitoring is available on z/OS only. This property is not shown if your project is connected to a non-z/OS TIBCO Object Service Broker installation.

If a copybook is not selected, the checkbox is disabled.

Load Specifies that the VSAM table is to be initialized with data.

Data cleansing The data-cleansing actions, if any, to be carried out by the Execution Environment. This property is optional.

Server ID If the table is to be accessed remotely via the TIBCO Service Gateway for Files, specify the server ID to be used; otherwise, leave blank.

Dataset type The VSAM data set type.

Ignore Specifies a value to identify the records to ignore in a KSDS data set. The Ignore field is defined and used as follows: the syntax is V; the length is 35; the value entered in the field is compared to the start of each record's key; if the key field is a string containing any unprintable characters, you can specify a hexadecimal string using X' as a prefix and a single quotation mark (') as a suffix. For example: X'hhh...hh'.

Records are processed as follows:

- If you request a sequence of records (a FORALL statement) or a non-specific record (a GET statement without a WHERE clause) and the ignore value matches the start of a record's key, that record is ignored and the next one is obtained.

If you request a specific record (a GET statement with a WHERE clause) or perform update operations, no records are ignored.

DDName Name of a JCL DD statement that defines the data set containing the data you want to access. Overrides both the File name property and the Name parameter. If you specify a name in this field, note the following:

- The DDname must already be allocated by the TIBCO Object Service Broker UI Execution Environment, using a JCL DD statement or the TSO ALLOCATE command.
- If you are accessing multiple VSAM tables from a single data set, the DDname in all the VSAM tables must be the same.

File name The fully qualified name of the VSAM data set that holds the data you want to access. The maximum record size is 31,744 bytes. The record length must be equal to or greater than the total length of the table fields. The File name associates the VSAM data set with this VSM table if you do not enter a value in the DDname field. The data set may be currently non-existent, but it must exist before the table is accessed.

If you are accessing multiple VSAM data sets with the same record layout, you can omit both the File name and DDname and instead specify the data set name as the first data parameter (it must have syntax C and a maximum length of 44).

Under CICS, define the VSAM data set using RDO (Resource Definition Online). You must provide the key length of the VSAM records in the CICS RDO definition.

If you specify both a File name and a parameter, the parameter is used. If you specify both a File name and a DDname, the DDname is used. If you specify both a parameter and a DDname, the DDname is used.

Table Parameters

You can define two types of table parameters:

Data Used to access external data in different data sets using the same table definition or to access multiple record types or multiple occurrences. Depending on the syntax and length you assign the data parameter, TIBCO Object Service Broker UI interprets it as follows:

- **If IDgen is N, Syntax is C, and Length is up to 44** – TIBCO Object Service Broker UI interprets the parameter value as the name of the data set.
- **If IDgen is Y, Syntax is any character, and Length is any length** – TIBCO Object Service Broker UI uses the parameters for reading multiple record types or multiple occurrences. The parameter definitions must match the primary key definition of the base table.

Location Used to access external data through a peer server associated with another Data Object Broker (remote node). If you do not need to access remote data, you can delete the parameter. If you always access the external file remotely, the node from which you request the access may have either a minimal or full definition.

- Minimal Definition – used when you always access data at a remote node. It consists of the following:
  - — The table name, which must be the same at both locations
  - — The location parameter, which must be the same at both locations. The name of the remote node where the full definition is located must be supplied in the Default field, Source field, or Source and Sourcename fields.

  The table type specified in a minimal definition need not match the table type of the full definition on the remote node.
- Full Definition – used when you can access data at either the local or the remote node. The table type of the full definition must match the data on the local node. For example, a full definition of type TDS used to access TDS data on the local node may also be used to access a VSAM table with the same name on a remote node.

For details on defining data and location parameters, see Definition of Table Parameters.

Table Fields

When defining a VSM table, the following applies:

- When defining fields, you can select external attributes (External Syntax, External Length and External Decimal) and UI will select reasonable defaults for internal attributes (Syntax, Length and Decimal) and vice versa. If these values are not satisfactory they can be re-selected manually.
- When you add a new field UI will generate a default value for Offset, based on the size and offset of preceding fields. If this value in not satisfactory it can be adjusted manually.

These fields define the external attributes and internal TIBCO Object Service Broker attributes for the primary key fields and data fields of the table.

If a copybook was selected using the Data Discovery wizard, fields may be selected directly from the data set member or file containing the copybook. The data set or file will be read each time Add from COBOL copybook... is selected. TIBCO Object Service Broker field names, external syntax, external length, external decimal and offset values will be generated from the copybook values.

Name Name of the field that must be unique within the table. You can use a name from another table; if you are moving data between this table and another table, giving fields the same name simplifies the process.

External Syntax External syntax for the VSAM field. The external syntaxes E and K are not valid for TIBCO Object Service Broker UI for z/OS.

External Length External length for the VSAM field.

External Decimal External number of decimal places for the VSAM field.

Offset The offset maps to the start of the external record relative to the VSAM field. The origin is zero. Overlaps are allowed for read-only tables and you do not need to define fillers, since the offset can be used to skip undefined locations in the row.

Key Type Specifies whether the VSAM fields are to be used in the primary key. You can select any field as the primary key, without respect to uniqueness of data. You can select up to 16 contiguous fields for a composite primary key, to a total maximum length of 127 bytes.

Unexpected results can occur if a field of syntax B or P is defined as a primary or secondary key field of a VSM table. The results occur because the two numeric syntaxes represent signed values, whereas VSAM key fields are always interpreted as unsigned values. Therefore, records can be placed out of sequence if the data contains both negative and positive values for the key fields. Records are ordered correctly in the following situations:

- The primary or secondary key field contains only negative or positive values.
- A non-key field of numeric syntax contains both negative and positive value and an ORDERED clause is used for selection.

Type The TIBCO Object Service Broker UI semantic data type of the field. The default is blank. You can specify any valid semantic data type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range

from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax The TIBCO Object Service Broker UI syntax of the field. You can specify any valid semantic type and syntax combination supported for the external syntax. For valid combinations, refer to *TIBCO Object Service Broker Programming in Rules*.

Length Length of the VSAM field. The data is padded or truncated as necessary.

Decimal Number of digits to appear to the right of the decimal point. The data is padded or truncated as necessary.

Order The ordering in which the occurrences in this field are sorted. Select **A – Ascending** or **D – Descending** or leave blank.

Required Specifies if a value is required for each occurrence in the table. Select **true** or **false** or leave blank.

Default Default value for the field, if no data is available.

Reference Name of a reference table. Used to check field values that are inserted or replaced. Having a reference table insures that only valid field values are stored in the field.

Global Field Name Name of the global field (read-only).

Field Controls

For details on the buttons for adding, moving, or deleting fields, see Parameter and Field Actions.

Event Rules

To manage the event rules for a VSM table, see Table Editor Event Rules.

Documentation

To manage the description and usage for a VSM table, see Documentation Page.

Addition of Table Reference to a Project

All table editors have the **Add to Project** button for adding a table reference to a project. The button displays in the upper right corner of the editor and is represented by the icon shown below.

Definition of Table Parameters

The following fields are used to define a parameter:

Name Name of the parameter. The order in which the names are entered is the order in which supplied values are processed. The value of the location parameter is provided by default, but can be modified.

Type Semantic data type of the parameter. Must be **I – Identifier** for the location parameter. Valid values: **<blank> – Typeless**: literals defined in TIBCO Object Service Broker rules and fields defined without a semantic data type; **C – Count**: integer numbers only (no fractions) used for integral count; **D – Date**: dates; you must include at least a year. Valid dates range from 0001/01/01 to 9999/12/31 inclusive; **I – Identifier**: unique identifier; **L – Logical**: information meeting Yes or No conditions; **Q – Quantity**: real numbers used for measurement; **S – String**: character string data.

Syntax Syntax of parameter. Valid values: **B** – binary; **C** – fixed-length character string; **P** – packed decimal. Must be **C** for the location parameter. For SUB tables with a source table of DB2 or SLK, type syntaxes **V**, **UN** or **RD** are also valid.

Length Length of the parameter value. The value is in bytes and valid values are determined by the syntax of the parameter. Each data parameter must have a non-empty value. Must be **16** for the location parameter.

Decimal Number of digits to the right of the decimal point. In most cases this is optional. It is relevant only for syntax P.

Class Class of parameter. Valid values: data parameter (**D**) or location parameter (**L**). For VSM and MAP tables, it can also be an address parameter (**A**) or a count parameter (**C**).

Reference (optional) Name of a reference table, used only if a table is to be referenced when a user is inserting or replacing a value in the parameter. You then ensure that the user enters valid values for the data parameter. This value is ignored for a minimal table definition.

Default (optional) Only used for parameters of location (L) class. Name for the default node to be used on data access. Even if a value is provided, when an access is made to the data, the location is determined using the order of evaluation described in "Order of Evaluation to Determine Location" in *TIBCO Object Service Broker Managing Data*.

Source (optional) Determines how the name of the node for the location parameter (class **L**) is determined. The value can be derived (**D**) from a functional rule named in the Source Name field or provided through the users' session options (**L**). Even if a value is provided, when an access is made to the data, the location is determined using the order of evaluation described in "Order of Evaluation to Determine Location" in *TIBCO Object Service Broker Managing Data*.

Source Name Name of the source rule used when the Source field is set to D. Must be a functional rule that returns the value for the node name. It must have an argument for the table name and arguments for each of the data and location parameters.

For additional details on table parameters, see *TIBCO Object Service Broker Programming in Rules* and *TIBCO Object Service Broker Managing Data*.

Parameter Controls

For details on the buttons for adding, moving, or deleting parameters, see Parameter and Field Actions on page 94.

Parameter and Field Actions

The following buttons add, copy, move, or delete the parameters and fields in the table editors.

Button	Description
Add	Adds a new parameter or field after the selected one.
Add from copybook...	Adds a new field from the COBOL copybook at the end of the table.
Add from Source...	Adds a new parameter or field from the source table after the selected parameter or field.
Copy	Copies the selected parameter or field.
Remove	Removes one or more selected parameter or field.
Move Up	Moves a parameter or field up in the list.
Move Down	Moves a parameter or field down in the list.

For IMP, EXP, MAP, VSM, and DAT tables, a COBOL copybook location is shown in the button area.

Even though this field is read-only, you can copy its content to the clipboard.

Additional Actions

In some cases, more actions are available. These are shown under the More Actions dropdown menu.

Menu	Action
Move Primary Keys to the Top	Consolidates the primary-key fields at the top of the table to allow saving of the definition. If the primary-key fields are not at the top of the definition, it cannot be saved. This action quickly resolves this error.
Recalculate Offsets	Recalculates all the offsets so that all the fields are positioned sequentially without gaps between them. This action is applicable for IMP, EXP, MAP and VSM tables.
Select Copybook...	Enables you to reselect a copybook associated with the table. See Chapter 4, Data Discovery.
Deselect Copybook	Removes the copybook association from the table. This action does not affect any other parts of the table.

Table Editor Event Rules

The event rule feature allows you to associate business rules and policies with the definition of a table. These rules are run whenever data in the table is manipulated. You can specify the following types of event rules:

- **Trigger** – causes additional processing to take place when a table is accessed. For example, it can be used to create an audit trail, or maintain a one-to-one relationship between two tables.
- **Validation** - used when the table is being modified. It checks the validity of modified values of fields in the table.

These rules run based on defined accesses. All the rules that apply to a specific access are executed in the order in which they are entered. You can specify as many rules as you need in any logical order. For additional details, see *TIBCO Object Service Broker Managing Data.*

Column	Description
Rule	Name of the event rule to be executed when the table is accessed.
Type	Type of event rule. Select T - Trigger or V - Validation. For import tables, only T is valid, and the rule must be a function, it cannot change the contents of the triggering row, and it cannot use the TRANSFERCALL statement. Nested triggers are possible.
Access	Type of data access that invokes the event rule. Only one access type can be specified for each entry of the Type field. Select D - Delete, G - General, I - Insert, R - Replace, or W - Write.

Event Rules Controls

Button	Description
Add	Adds a new event rule after the selected rule.
Copy	Copies the selected event rule.
Remove	Removes one or more selected event rules.
Move Up	Moves an event rule up in the list.
Move Down	Moves an event rule down in the list.

Documentation Page

The Documentation Page is preset in the Rule editor, Table editor, Table Data browser and editor. This page is used for metadata that facilitates the use, management, and understanding of these objects. This page is read-only for Table Data browser and editor.

Attributes

Modified Date the object was last modified (read-only).

Modifier Id of the user who last modified the object (read-only).

Created Date the object was created (read-only).

Author Id of the user who created the object (read-only).

Unit Name for a group of related objects, of which this object is a member.

Keywords Significant or descriptive words about the object.

Summary Summary of the object's purpose, use, or characteristics.

Description This space provides for notes, comments, or any other relevant text. The description can have embedded formatting commands that alter the appearance of the text. See Formatting Commands below.

Formatting Commands

Usage	Purpose
.AD #	Set left margin # spaces from edge
.BC CHARS	Char becomes the bullet characters
.BM #	Bottom margin gets # lines
:BOX	Start a box
.BR	Break
.BREAK	Break

Usage	Purpose
.BU{.TEXT}	Place bullet before text
.CC{.TEXT}	Center and uppercase text
.CE{.TEXT}	Center text
.CO {ON¦OFF}	.co off will pass lines through
.DATE {FORMAT}	Insert today's date here in format
:DD{.TEXT}	Definition for list
.DL	Start a definition list
.DOUBLE	Double space lines
:DT{.TEXT}	Text is a term for definition list
:EBOX	End a box
:EDL	End definition list
:EOL	End ordered list
:EUL	End unordered list
.FO {ON|OFF}	Turn formatting on or off
.FORMAT {ON|OFF}	Turn formatting on or off
.HEADING1{.TEXT}	Level 1 heading
.HEADING2{.TEXT}	Level 2 heading
.HEADING3{.TEXT}	Level 3 heading
:H1{.TEXT}	Level 1 heading
:H2{.TEXT}	Level 2 heading
:H3{.TEXT}	Level 3 heading
.IMBED <TABLENAME>	Use <tablename> as source of text
.IN #	Indent left margin # spaces

Usage	Purpose
.INDENT #	Indent left margin # spaces
.ITEM{.TEXT}	Begin a list item
.JU {ON|OFF}	Turn right justification on or off
.JUSTIFY {ON|OFF}	Turn right justification on or off
.LI{.TEXT}	Item for ordered or unordered list
.LIST CHAR	Begin a list of any kind
.LISTEND	End any list
.LL #	Line length is # characters
.SP #	Space lines #
:OL	Start ordered list
:P{.TEXT}	Start a paragraph
.PA	Start a new page
.PAGE	Start a new page
.PARA{.TEXT}	Start a new paragraph
.PL #	Page has # lines including margins
.PN {ON|OFF|#}	Page number is #, or is on or off
.PP{.TEXT}	Start a new paragraph
.SETUP NAME	Use the named setup
.SINGLE	Single space lines
.SK #	Skip # lines
.SKIP #	Skip # lines
.SP #	Skip # lines
.TABLE {F1,F2,..,FN}	Print table into output

Usage	Purpose
.TERM{.TEXT}	Text is a term for definition list
.TM #	Top margin has # lines
:UL	Start an unordered list

Documentation Page Controls

Button	Description
View	Displays the formatted description text.
Edit	Displays an editable text box where the text and formatting commands can be entered.

Editing of Tables That Contain Data

When a table contains data, certain changes to the definition are not allowed. In that case the Table editor will not permit such changes.

In some situations, the UI is unable to determine if a table has data. For example, the data segment might be off line. In such cases, a warning message is displayed, but the Table editor will not restrict the changes to the table definition and you will be making changes at your own risk.

In general, you are safe if you are NOT shortening a field length, changing a field or parameter syntax and decimal length, changing a parameter or key length, adding new fields in between existing ones, adding new parameters, deleting existing fields or parameters, or modifying the "key" attribute of any field.

Care must be taken when performing these modifications.

Chapter 4 Data Discovery

You can build an IMP, EXP, VSM, MAP, or DAT table with a COBOL copybook, which enables the creation of table fields that match the data types and binary locations of copybook items.

Creation of Table Fields

If a table has a copybook selected, you can click the **Add from copybook...** button to quickly create a field based on a copybook item.

To enable that, first associate a table with a copybook. You can select a copybook in either of these two ways:

- When creating a new table with the New Table wizard, you can navigate to a copybook from the second screen.
- For tables that already exist, click **More Actions**, then click **Select Copybook...**

The controls in the wizard and the dialog box are the same. See the table below.

For DAT tables in the New Table wizard, the copybook-selection step is mandatory. For IMP, EXP, VSM, or MAP tables, that step is optional.

Field or Column	Description
Dataset(directory) filter	Filter string for data set names. Type the filter string and click **Find**. On Open Systems, type the name of a directory.
Datasets(directories)	On z/OS, data sets that match the specified filter. On Open Systems, a list of subdirectories. Select the desired data set or directory.
Members(files)	On z/OS, data set members in the selected data set. On Open Systems, a list of files in the selected directory. Select the member or file that contains the desired data layout information for the table definition.
Source	The source text of the selected member.

To move between columns, click **Tab** or **Alt-Tab**.

If a copybook has been selected, the Table editor activates the **Add from copybook...** button. The copybook is then read each time you click that button. Additionally, when monitoring is selected for a table, the current member statistics are compared with the saved statistics for the table and a warning message is issued if there is a difference.

Messages

Messages on monitoring might appear. Click **Yes** to monitor the data member for changes. Otherwise, click **No**. Monitoring is available on z/OS only.

Chapter 5 Editors

This chapter describes the editors (other than table editors) provided by the TIBCO Object Service Broker UI.

Topics

Application Editor

This editor is used to manage the properties of an application.

Application Properties

These properties are searchable; you can search for an application based on one property or a combination of these properties.

Title Title of the application (20 character maximum).

Description Description of the application's purpose, use, or characteristics (60 character maximum).

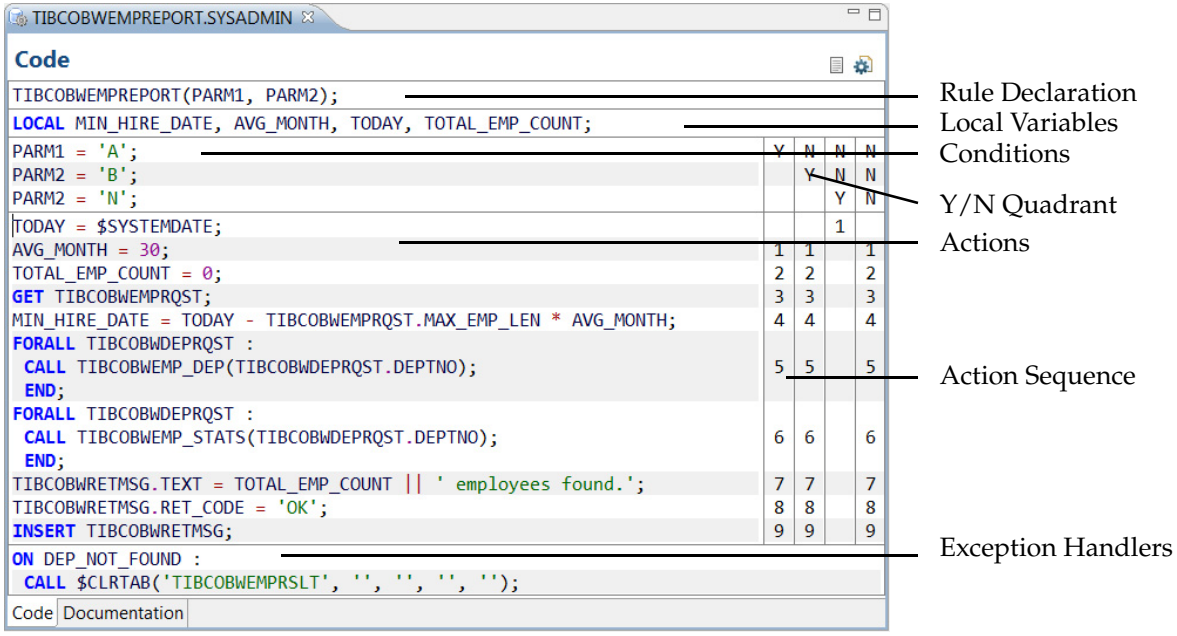
Unit Name for a group of related objects, of which this application is a member (8 character maximum).

Rule Editor

This editor is used to manage rules. For information on the rules language, see Appendix A, Rules Language Reference, on page 187.

Code Page

The Code page is used to enter the code for the rule you are editing. The page has the Add to Project button to add a rule reference to a project.



Rule Declaration This area contains the rule name and parameters. Comma-separated parameters are enclosed in parentheses. The rule name, the parentheses, and the semicolon are not editable.

Local Variables Enter local variables after the space following the word LOCAL. The word LOCAL and the semicolon at the end are required rule syntax and cannot be modified.

Conditions Enter rule conditions. Each condition resides on a separate line, terminated with a semicolon. A rule can have up to 6 conditions.

Actions This section contains rule actions. Each action has corresponding rows of **Y/N Quadrant** and **Action Sequence Numbers**.

Actions executed when a certain rule condition is true is being determined by the Action Sequence Number area. If an action is to be executed when a particular condition is met, the corresponding box in the Action Sequence Number area contains a number. The value of the number indicates the sequence of execution for a condition and is not editable.

Clicking a corresponding area in the Action Sequence Number area toggles the number, thus making the corresponding action executable or not under a condition.

The Action Sequence Number area always contains an extra column filled with 'N' letters. This sequence is executed when none of the conditions are met. When there are no conditions in the rule, the Conditions area should be left empty.

Note: You can navigate to the Action Sequence Numbers area via the Tab and Shift-tab keys, and toggle the Action Sequence Number via the Space or Enter keys. Also, you can use the arrow keys for navigation within the Action Sequence Numbers area.

Y/N Quadrant An information area that cannot be modified. Clicking on a column that contains the Y and N letters will highlight all actions executable for the condition. Click again to remove the highlighting.

Exception Handlers This area contains exception handling code.

Performing Actions on Referenced Objects

The Rule editor recognizes the use of specific objects in the rule code and provides specific actions for the object at the cursor. For example, if you have the following statement in a rule:

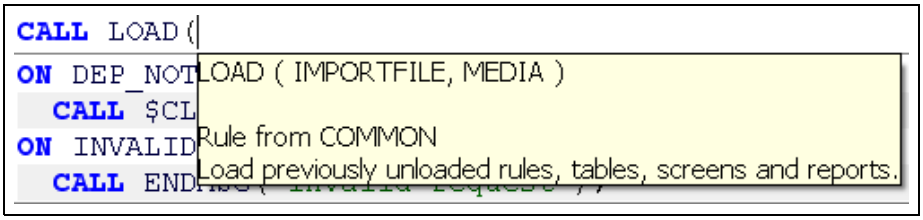
```
CALL MYRULE;
```

you can position the cursor anywhere in `MYRULE`, right-click, and invoke any of the specific menu actions, such as Open or Add to Project.

Documentation Page To manage the description and usage for a rule, see Documentation Page.

Object Pop-ups

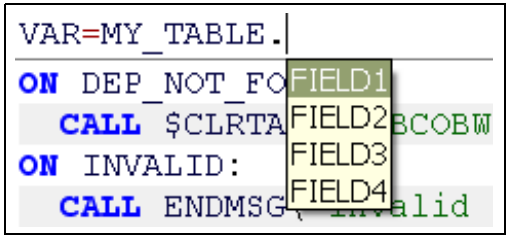
When you type a recognizable rule name, the rule info pop-up displays:



You can re-display the pop-up by placing the cursor on a rule name and pressing Ctrl+Space.

Note: The same pop-up displays when you type the name of a table in a table access statement.

When you type a table name followed by a dot, a list of table fields displays:



You can select a field and it will be inserted in the rule.

View a Rule in a Text Window

You can view a rule as text. Click the button, located at the top right of the screen, and a text window containing the rule opens. The text in window is read only. You can use standard Eclipse menus under File, including Save As... and Print. You can also copy the text into the clipboard and paste in another application such as an e-mail client.

Rules Language Reference

See Appendix A, Rules Language Reference, on page 187.

Table Data Browser

The Table Data browser allows you to browse table data.

Browse Page

This page shows the table data.

Button or Label	Description
Table specification	Shows the name of the table with selected parameters. See Table Data Selector Dialog on page 154 for more details on parameter selection.
Selection criteria	If the selection criteria is specified in the Table Data Selector, it will be displayed above the table data.
Local library	Displays the local library selected in the Table Data Selector dialog. See Local Library in Table Data Browser and Editor on page 114.
 Set data filter	Allows you to re-specify the table parameters and selection criteria.
 Reload data	Reloads data from the table and positions selection of the first occurrence.
 Add to Project	Adds a table reference to a project.
Table data	Table data area.

Documentation Page

This page allows you to view the metadata for a table; for details, see Documentation Page. To modify the description and usage, use the table editor.

How Data is Loaded

Data is loaded on demand, in portions, as you scroll down. The scrollbar periodically resizes as scrolling takes place.

The Table Data browser starts a dedicated session and a transaction, and keeps those open until you close the editor. The Table Data browser places no locks on data. Table definition lock, however, is placed, so the definition cannot be modified or deleted while a Table Data browser is open.

Occurrences that have already loaded stay in the UI buffer, and it is possible that if another person modifies the data, your view of the data will become outdated. Use the Refresh button to discard all loaded occurrences and reload the first visible page of data.

Table Data Editor

The Table Data Editor allows you to edit table data.

The UI uses the Table Data Editor for all editable table types, except for DAT. When you edit the data of a DAT table, the UI opens a 3270-based data editor.

Edit Page

This page shows the table data.

Button or Label	Description
Table specification	Shows the name of the table with selected parameters. See Table Data Selector Dialog on page 154 for more details on parameter selection.
Selection criteria	If the selection criteria is specified in Table Data Selector it will be displayed above the table data.
Local library	Displays the local library selected in the Table Data Selector dialog. See Local Library in Table Data Browser and Editor on page 114.
Set data filter	Allows you to re-specify the table parameters and selection criteria.
Reload data	Reloads data from the table and positions selection of the first occurrence.
Add to Project	Adds a table reference to a project.
Table data	Table data area with in-place editing capability.
Add	Adds a new occurrence
Remove	Removes selected occurrences
Copy	Copies selected occurrences.

Documentation Page

This page allows you to view the metadata for a table; for details, see Documentation Page. To modify the description and usage, use the table editor.

How Data is Loaded

Data is loaded on demand, in portions, as you scroll down. The scrollbar periodically resizes as scrolling takes place.

The Table Data editor starts a dedicated session transaction and keeps those until you close the editor. Initially, the Table Data editor places shared lock on data and upgrades the lock to exclusive when changes are saved, according to general TIBCO Object Service Broker principles.

Editing and Saving Data

The Table Data editor keeps the changes in the UI buffer until they are saved. Only then are they transmitted to the TIBCO Object Service Broker back-end.

When data is saved, the Table Data editor commits the data and keeps its transaction active; therefore, all locks persist/upgrade until you close the editor.

The Table Data editor limits the amount of changes it allows, not to exceed the intent list. This ensures that there are no intermediate commits during saving. When the limit is reached, you must save your changes to be able to make more changes.

If a text field in your table contains control characters, such as 'carriage return' and 'line feed,' the editing behavior may be unexpected. For example, Eclipse text components may expand 'carriage return' characters into 'line feed' and 'carriage return' pairs. Generally, editing fields with such values is not recommended.

Local Library in Table Data Browser and Editor

Each instance of a Table Data Browser or Editor uses a separate TIBCO Object Service Broker session. This session is started with the SEARCH parameter set to the value **L**, which means that trigger and validation rules for the table will be searched for in local, installation and system libraries (in that order).

The Table Data Selector dialog allows you to specify a local library for the above process.

Transaction Editor

This editor is used to manage the properties, linked documents, user data, and input/output tables of transactions.

Properties – Identification

Application The application that contains the transaction (read-only).

Title Title of the transaction (20 character maximum).

Description Description of the transaction (60 character maximum).

Unit An arbitrary name used to organize related objects into groups, of which this transaction is a member (8 character maximum).

TIBCO BusinessWorks compatible Enable the transaction to be called by TIBCO BusinessWorks. In this mode, transactions require additional information and the transaction editor will show two additional pages: Inputs and User Data. Note that Linked Documents are not applicable in this mode and therefore the Linked Document page is hidden.

For details, refer to the TIBCO BusinessWorks documentation that is included in the TIBCO Object Service Broker documentation set.

Properties – Runtime Attributes

Initial rule Name of a rule to be executed at the beginning of the transaction.

Post process rule Name of a rule to be executed at the end of the transaction. The rule is executed at the very end of transaction processing, after any linked documents have been processed.

Update mode Run the transaction in update mode, which allows the transaction to make changes to the database. When not selected, the transaction runs in browse mode, which does not permit database changes.

Dump diagnostic to log Record detailed trace information for each execution of this transaction to Transaction Log. These logs can be viewed in the Transaction Logs view. Logs are arranged by the date they were generated.

Add to Project Button

Adds a transaction reference to a project.

Transaction Editor Linked Documents Page

After the rules processing for the transaction is complete, XML documents in the Linked Documents list are processed in top-down order.

You can add XML documents to the list by dragging XML document names and dropping then in the list view.

Note that this page is not present if the TIBCO BusinessWorks compatible checkbox (Properties page, Identification) is checked.

XML Document Controls

Button	Description
Move Up	Moves the selected XML document up in the list.
Move Down	Moves the selected XML document down in the list.
Remove	Removes one or more selected XML documents.

Transaction Editor Input Tables Page

This page lists the input tables to be used by TIBCO BusinessWorks when calling this transaction. You cannot use tables with data parameters as input.

To add input tables to the list:

1. In the Tables view or OSB Projects view, select the tables you want to add.
2. Drag the tables into the Input Tables list.

To remove input tables from the list:

1. Select the tables you want to remove from the Input Tables list.
2. Click **Remove**.

Transaction Editor Input User Data Page

This page lists the names of user data to be used by TIBCO BusinessWorks when calling this transaction.

To add a name to the list:

1. Click **Add**.
2. Modify the user data name in the Name text field.

To remove a name from the list:

1. Select the name you want to remove from the Name list. You can select multiple names if desired.
2. Click **Remove**.

Transaction Editor Output Tables Page

This page lists the output tables to be used by TIBCO BusinessWorks when calling this transaction.

Details (of selected table in the list)

Table Name of the output table.

Build Rule Name of a rule that is run for this table when the transaction is invoked.

Key predicate The selection criteria used to select data from the table. See Parameters, Key Predicate and Ordering Syntax on page 131.

Ordering The sort order criteria used to sort the data from the table.

Parameters The criteria that determines the parameter value required to access a parameterized table. If the selected output table is a parameterized table, you can use this field to specify the table instance to be returned. See Parameters, Key Predicate and Ordering Syntax on page 131.

To add an output table to the list:

1. In the Tables view or OSB Projects view, select the table you want to add.
2. Drag the table into the Table list.

To remove an output table from the list:

1. Select the table you want to remove from the Table list.
2. Click **Remove**.

XML Document Editor

This editor is used to manage the properties, child documents, and tables of XML documents.

Identification

Title Title of the XML document (20 character maximum).

Description Description of the XML document (60 character maximum).

Unit An arbitrary name used to organize related objects into groups, of which this XML document is a member (8 character maximum)

Runtime Attributes

Root name Name of the root element in the XML document.

Type The style of XML document to be produced or consumed. When there are child documents, the value in the parent document takes precedence.

Validate document Validate the XML document against its DTD or XML schema.

Build header Insert a default XML declaration at the head of the document, such as the following:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Applicable only if Type is set to XML.

Recurse When processing this document, recursively process any child documents as well. Applicable only if Type is set to XML.

Build DTD Generate an inline DTD when the document is processed. Applicable only if Type is set to XML.

Processing Options

Regular expression for... A valid regular expression indicating the parts of the XML document to parse. Applicable only if Type is set to XML on the Properties tab.

Production rules

Specify the rules to run when this XML document definition is being used to produce an XML document.

When a parent XML document contains child XML documents, the parent's rules are run once for the parent, and the child's rules are run once for each row in the parent table.

Pre-process rule Name of the rule that is executed at the beginning of processing, before the first table data item is read.

Post-process rule Name of a rule that is executed at the end of processing, after the last table data item is read.

Consumption rules

Specify the rules to run when this XML document definition is being used to consume an XML document.

When a parent XML document contains child XML documents, the parent's rules are run once for the parent, and the child's rules are run once for each row in the parent table.

Pre-process rule Name of the rule that is executed at the beginning of processing, when the root element for the table is encountered.

Post-process rule Name of a rule that is executed at the end of processing of the child items in the XML document table, but before the table data is inserted.

Add to Project Button

Adds an XML document reference to a project.

XML Document Editor Child Documents Page

This page is used to specify the data parameter value and data key value required to access and select data from the selected XML child document. After the rules processing for the parent XML document is complete, XML documents in the Child Documents list are processed in top-down order.

Properties (of the selected document in the list)

Document Name of the XML child document.

Parameters The criteria that defines the parameters required to access a parameterized data table. See Parameters, Key Predicate and Ordering Syntax on page 131.

Key predicate The selection criteria used to select data from the data table. See Parameters, Key Predicate and Ordering Syntax on page 131.

XML Child Document Controls

Button	Description
Move Up	Moves the selected XML document up in the list.
Move Down	Moves the selected XML document down in the list.
Remove	Removes one or more selected XML documents.

XML Document Editor Tables Page

This page lists the output tables to be used by TIBCO BusinessWorks when calling this transaction.

Properties (of the selected table in the list)

Table Name of the XML document table.

Root Name Name of the root element for the XML document table. Unlike the root element of the XML document, the root table element can occur in the document multiple times, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rootelement>
    <roottableelement>
      <xmldata>
      </xmldata>
    </roottableelement>
    <roottableelement>
      <xmldata>
      </xmldata>
    </roottableelement>
  </rootelement>
```

When producing XML, all the XML definitions in the field map are subordinate to the root table element. When consuming XML, this root name is used to determine which XML document or XML document table definition is to be used to map the parsed XML statements.

Map name Name of the XML field map associated with the XML document table.

To associate an existing field map with the XML document table:

In the XML Field Map view, select the field map you want to associate with the XML document table and drag it into the Map name field.

To open the specified XML field map:

Right-click the name in the Map name field and select **Open**.

To create a new XML field map that is associated with the XML document table:

Use the New XML Field Map Wizard.

You can invoke the New XML Field Map wizard by typing a name into the Map name field and opening it via the right-click menu. If the XML field map with that name does not exist, the New XML Field Map wizard will be displayed.

Pre-process consumption rule Name of the rule that is executed at the beginning of processing, when the root element for the table is encountered.

Post-process consumption rule Name of a rule that is executed at the end of processing of the child items in the XML document table, but before the table data is inserted.

Pre-process production rule Name of the rule that is executed at the beginning of processing, before the first table data item is read.

Post-process production rule Name of a rule that is executed at the end of processing, after the last table data item is read.

Generate empty When generating documents of type XML, specifies whether an attribute or element should be generated if the mapped data field contains a null. By default, no attribute or element is generated.

Null after update When consuming inbound XML, each data item is placed in a table buffer before being added to the database at the closing element. If selected, specifies that the buffer is to be cleared before processing of the next element begins. If not selected, and the next element does not contain any data items, the values added to the database will be the same as for the previous XML data element.

Parameters The criteria that is used to determine the parameter values required to access a parameterized data table. See Parameters, Key Predicate and Ordering Syntax on page 131.

Key predicate The selection criteria used to select data from the data table. See Parameters, Key Predicate and Ordering Syntax on page 131.

Ordering The sort order criteria used to sort the data from the table. See Parameters, Key Predicate and Ordering Syntax on page 131.

Updating the XML Document Table

These properties control how and when the XML document table is updated on XML document consumption.

Update at end of element When you have defined nested XML documents that contain data for the same table and row, select this option to prevent the partially constructed row from being inserted at the end of the child document element.

Replace existing data Allow existing data in the XML document table to be overwritten.

Commit point

The following options are available for setting the commit point:

- **At COMMITLIMIT** – Updates committed to the XML document table when a COMMITLIMIT occurs. A COMMITLIMIT occurs when the limit on the number of updates between synchronization points has been reached.
- **End of root** – Updates committed at the end of each child document root element. Caution: do not select this option for child documents.
- **After N roots** – Updates committed at the end of the specified number of child document root elements. Caution: do not select this option for child documents.
- **None** – Updates committed at the end of the parent document root element.

  Note: For large documents, COMMITLIMIT error may occur. In this case, all data changes are rolled back and data integrity is not compromised.

This example shows when changes are committed for the above settings:

```
<rootelement>
  <childrootelement>
    ...
  </childrootelement> <-- End of Root
  <childrootelement>
    ...
  </childrootelement> <-- After 2 Roots
</rootelement>  <-- None
```

Using **End of root** or **After N roots** for child documents may cause significant data integrity issues in the event the commit process fails. It is strongly recommended that child document definitions use only **At COMMITLIMIT** or **None**.

XML Tables Controls

Button	Description
Move Up	Moves the selected XML document up in the list.
Move Down	Moves the selected XML document down in the list.
Remove	Removes one or more selected XML documents.

To add an output table to the list:

1. In the Tables view or OSB Projects view, select the table you want to add.
2. Drag the table into the Table list.

To remove an output table from the list:

1. Select the table you want to remove from the Table list.
2. Click **Remove**.

XML Document Editor Tree Page

This page shows child XML documents and all their children. If a document recursion is detected it will be indicated by a warning icon on the left side of the document. You can perform tasks on any document shown on this page using the documents task menu.

XML Field Map Editor

This editor is used to manage the properties of XML field maps.

Identification

Modified Date the field map was last modified (read-only).

Modifier Id of the user who last modified the field map (read-only).

Created Date the field map was created (read-only).

Author Id of the user who created the field map (read-only).

Title Title of the field map (20 character maximum).

Description Description of the field map (60 character maximum).

Unit Name for a group of related objects, of which this field map is a member (8 character maximum).

Add to Project Button

Adds an XML filed map reference to a project.

XML Field Map Editor Field Maps Page

An XML field map has two complementary purposes. First, it allows the definition of the data to be included as part of the production of an XML document, and to determine how this data is represented in the document. Second, it defines the way that data parsed from an XML document is processed and where the data is placed.

Table Fields

Field name Name of the data table field that is being mapped.

Type Type of the data field. Each data field can be represented in the outbound XML document in the following ways: as an element, as an attribute of the table root name, as an attribute of another element, or as an empty element.

Root name The value that identifies the field in the XML document. If not specified, the value provided in the Field name field is used.

Group name Name of an XML element under which related fields are grouped.

For example, when producing an XML document from the customer table you want to include details about the customer's name and age. By simply including the three fields in the field map, the following document is produced:

```
<Customer>
  <FirstName>some data</FirstName>
  <SurName>some more data</SurName>
  <Age>35</Age>
</Customer>
```

However, the name fields should be grouped together under the element "Name". By specifying a Group Name value of "Name" for both fields, the following document is produced:

```
<Customer>
  <Name>
    <FirstName>some data</FirstName>
    <SurName>some more data</SurName>
  </Name>
  <Age>35</Age>
</Customer>
```

Output format rule Name of a rule that formats the data contained in the field when producing the XML document. On entry to the rule, the local variable NEW_FIELD_VALUE contains the value of the field identified by the Field name field. The rule needs to set this local variable to a string that is the desired output value. For example, on rule entry NEW_FIELD_VALUE is set to "12/01/98" and on rule exit it is set to "1st December 1998".

Input format rule Name of a rule that formats the data parsed from the XML document being consumed. On entry to the rule, the local variable NEW_FIELD_VALUE contains the value contained in the XML document. The rule needs to set this local variable to a value that is placed into the table field. For example, on rule entry NEW_FIELD_VALUE is set to "1st December 1998" and on rule exit it is set to "12/01/98".

Usage Type of XML processing for which the field is used. For example, for production only, for consumption only, or for both production and consumption. By default, fields are used for both the production or consumption of XML. However, you may want to override the default usage setting. For example, it is possible that you may want to ignore an XML entity in an inbound document, but to have the XML entity included when the same document is produced. Select **Input only**, **Output only** or leave blank.

Table override Alternate location for the inbound data to be placed.

Empty element name Overrides the default attribute name of "value" with the specified value. For example, when a field is specified as type Empty Element in the Type field, the following XML entities are produced:

```
<FieldName value="fieldvalue" />
```

Specifying a name of "mydata" in the Empty Element Name map field produces the following:

```
<FieldName mydata="fieldvalue" />
```

XML Document Controls

Button	Description
Add from table...	Adds a new field from a table specified in the Table field of the XML Document, in the same row where this field map is used in the Map Name field.
Move Up	Moves a row/item up in the list.
Move Down	Moves a row/item down in the list.
Add	Adds a new field after the selected field.
Remove	Removes one or more selected fields.

Parameters, Key Predicate and Ordering Syntax

Transactions and XML Documents use common syntaxes for Parameters, Key Predicate and Ordering.

Parameters Syntax

The parameter value is specified to indicate particular table data parameters. It is provided in one or more comma-separated values that represent either the data value itself, or a reference to the data value.

The following example specifies two dynamic substitutions, `{ARG1}` (a numeric value) and `'{TABLEA}.{FIELDA}'` (a string value):

```
'A Value', {ARG1}, '{TABLEA}.{FIELDA}'
```

If the value to be substituted is a string value, the expression must be enclosed in single quotation marks.

Syntax in BNF Notation

<parm> ::=
<parm value list>
<parm value list> ::=
<parm_value{,parm_value}>
<parm_value> ::=
<passed user data name>
<table reference>
<data value>
<passed user data name> ::=
<{user data name name}>
<table reference> ::=
<{tablename}.{fieldname}>
<data value>::=
<Numeric Value>
<Quoted String>
<Quoted String>::=
<'string'>

Key Predicate Syntax

The key predicate provides table access predicates in a syntax very similar to the syntax used for rules, but without the table specification, the WHERE, or any ordering.

The following example specifies two dynamic substitutions, {ARG1} (a numeric value) and '{TABLEA}.{FIELDA}' (a string value):

```
FIELD1 = 'A Value' AND FIELD2 > {ARG1} OR
FIELD3 = '{TABLEA}.{FIELDA}'
```

If the value to be substituted is a string value, the expression must be enclosed in single quotation marks.

Syntax in BNF Notation

<key predicate> ::=
<where not expression> {<logical operator>
<where not expression>}
<where not expression> ::=
[<not>] <where expression>
<where expression> ::=
<where relation>
(<where predicate>)
<where relation> ::=
<field reference> <relational operator>
<where expression>
<where expression> ::=
[<unary operator>] <where expression term>
{<add operator> <where expression term>}
<where expression term> ::=
<where expression factor> {<multiplication operator>
<where expression factor>}
<where expression factor> ::=
<where expression primary> [<exponent operator>
<where expression primary>]
<where expression primary> ::=
<passed argument name>
<table reference>
<data value>
<passed argument name> ::=
<{argument name}>
<table reference> ::=
<{tablename}.{fieldname}>
<data value>::=
<Numeric Value>
<Quoted String>
<Quoted String>::=
<'string'>

Ordering Syntax

Ordering specifies the sort order criteria that is used to sort the data from the source data table. A sort order specification has the following form:

```
SORTORDER FIELDNAME & ORDERED SORTORDER FIELDNAME ...
```

where SORTORDER is ASC or ASCENDING, or DESC or DESCENDING. For example:

```
DESC NAME & ORDERED ASC ADDRESS & ORDERED DESC NUMBER
```

Note: Dynamic substitution using the {} syntax can be used for sort order specifications, just as it is used for Parameters and Key Predicates. See above.

TN3270 Editors

TN3270 editors are used for the following tasks:

- As editors for Libraries, Reports, Screens, and Object Sets. When the Open task menu item is selected for those objects, a TN3270 editor for that object is opened.
- To allow access to the traditional TIBCO Object Service Broker Text Workbench. Text Workbench offers a large number of tools.

You can open an unlimited number of TN3270 editors. However, a single TIBCO Object Service Broker can only support a limited number of open sessions. It is recommended that you refrain from opening more than a few TN3270 Text Workbench editors for the same project.

Within a 3270 editor, use F1 to open a help screen.

TN3270 Object Set Editor

Refer to TN3270 Editors on page 134.

TN3270 Report Editor

Refer to TN3270 Editors on page 134.

TN3270 Screen Editor

Refer to TN3270 Editors on page 134.

TN3270 Table Editor

Refer to TN3270 Editors on page 134.

TN3270 Text Workbench Editor

Refer to TN3270 Editors on page 134.

TN3270 Library Editor

Refer to TN3270 Editors, page 134.

Chapter 6 Dialogs

This chapter describes the dialogs provided by the TIBCO Object Service Broker UI.

Topics

Open Object Dialog

The Open Object dialog provides a quick way to invoke a select set of actions on an object. This dialog has a keyboard shortcut Ctrl+O assigned to it by default.

Depending on other plug-ins installed in Eclipse, there could be more than one action assigned to a particular shortcut. In that case, when the shortcut is used, Eclipse shows a menu with actions assigned to the shortcut.

To resolve shortcut conflicts, access **Window > Preferences** to bring up the Preferences dialog. Select **General > Keys**. Locate the action named "Open Object" and modify the binding.

The Open Object dialog will not be displayed if OSB Projects View is closed.

Dialog Properties

Project name Displays the TIBCO Object Service Broker project selected in OSB Projects View. Non-modifiable.

Name Name of the object. The drop-down list contains a history of object names previously entered.

Type Drop-down list of TIBCO Object Service Broker object types.

Open from library This group is only displayed when the value of **Type** is "Rule." It has the following fields:

- **Library** (text box) – specify the library for the rule. You can use **Browse...** to selected a library from the list of existing libraries.
- **Search installation and system libraries**. If this checkbox is checked, the rule will be searched for first in the library specified in the **Library** text box, then in installation and system libraries. If unchecked, only the library specified in the **Library** text box is searched.

Note: Search installation and system libraries has no effect if the **Run As** action is selected.

Buttons

The buttons on the bottom of the dialog are used for invoking a specific action for the object. The set of buttons depends on the value selected in **Type** field, except **Cancel**, which closes the dialog.

Button	Selected Type Value	Action
Open	Any selection	Opens an editor for the object.
Run As	Rule, Transaction, XML Document	Opens Run dialog.
Edit Data	Table	Opens a Table Data Editor.
Browse Data	Table	Opens a Table Data Browser.

History

The **Name** field drop-down list contains a history of the objects you have accessed via this dialog. When you select an entry from the list, the **Type** field also changes to the type of the selected object. For rules, contents of Open from library group are also restored to the values used when that rule was accessed.

Note that the history is kept per project. Also note that the history does not persist and will be cleared when a project is closed or Eclipse is shutdown.

Paste Library Dialog

You use the Paste Library dialog when you make a copy of an existing library.

Dialog Properties

New name Name of the copied library.

Copy rules If selected, copies the rules from the original library into the copied library.

Using the Dialog

To use this dialog, perform the following:

1. In the **New name** field, type a name for the copy of the original library.
2. Select **Copy rules** if you wish to include the rules from the original library in the copy.
3. Select **OK** to create the copy; select **Cancel** to close the dialog without creating the copy.

Rename Object Dialog

You use the Rename Object dialog when you want to change the name a TIBCO Object Service Broker UI object (such as an Object Set or XML Document).

Dialog Properties

New name The new name for the TIBCO Object Service Broker UI object.

Using the Dialog

1. In the **New name** field, type the new name for the TIBCO Object Service Broker UI object.
2. Select **OK** to change the name; select **Cancel** to close the dialog without changing the name.

Run Rule Dialog

You use the Run Rule dialog to run a rule.

Dialog Properties

Name Name of the Run configuration that contains this rule. If multiple configurations exist, you can select the desired configuration using the dropdown. When you re-select the configuration the rest of the dialog is updated to reflect the settings from that configuration.

Click **Run** to save any modifications back to the configuration.

Local library Local library name. To locate the rule, a search in local, installation and system libraries will be performed. Use this field to specify local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

The rule may no longer be present in the selected library. If not present, a rule with the same name but from the installation or system library will be run. If the search cannot locate the rule by name, an error occurs.

Rule Name of the rule. You can use the **Browse...** button to select a rule from the list of rules added to the project. Note that only rules from the library specified in **Local library** field are displayed.

Arguments Arguments used when you run the rule. Arguments that have been defined for the rule appear in the **Name** column.

Update mode If selected, runs the rule in update mode, which allows the rule to perform updates to persistent database tables. If not selected, the rule runs in browse mode, and any attempt by the rule to perform an update to a persistent database table will cause the rule to fail.

Note: In browse mode, EXP and VSM tables as well as memory-resident tables can be updated.

Test mode If selected, runs the rule in test mode. When a rule attempts to perform updates to persistent database tables, it is allowed to run successfully, but the updates are discarded.

Note: Most accesses to external databases are not affected by test mode, and changes to the databases are persistent. Also, in test mode, EXP and VSM tables are updated.

Profiling If selected, collects information about executing the rule (profile).

Data file name The back-end system file name to store profiling information.

Using the Dialog

To use this dialog, perform the following:

1. In the **Local library** field, specify the name of the library (if not already populated).
2. In the **Rule** field, specify the name of the rule (if not already populated).
3. Specify values (in the **Value** column) for the arguments displayed in the **Name** column.
4. Select **Update mode** if desired.
5. Select **Test mode** if desired.
6. Select **Profiling** to get a rule profile. Provide a data file name for Profiler.
7. Select **Run** to run; select **Cancel** to close the dialog without running.

Run Transaction Dialog

You use the Run Transaction dialog to run a TIBCO Object Service Broker UI transaction.

Dialog Properties

Name Name of the Run configuration that contains this transaction. If multiple configurations exist, you can select the desired configuration using the dropdown. When you re-select the configuration the rest of the dialog is updated to reflect the settings from that configuration.

Click **Run** to save any modifications back to the configuration.

Local library Local library name. During transaction execution rules will be searched for in local, installation and system libraries. Use this field to specify the local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

Transaction Name of the transaction. You can use the **Browse...** button to select a transaction from the list of transactions added to the project.

User Data Data to be used when you run the transaction. The data consists of Name – Value pairs that you specify using the **Add** button.

Using the Dialog

To use this dialog, perform the following:

1. In the **Local library** field, specify the name of the library (if not already populated).
2. In the **Transaction** field, specify the name of the transaction (if not already populated).
3. Specify user data if desired by selecting **Add** and typing the Name–Value pair into the table. Repeat as necessary, or select **Remove** to delete an existing Name–Value pair.
4. Select **Run** to run; select **Cancel** to close the dialog without running.

Run XML Document Dialog

You use the Run XML Document dialog to run an XML document

Dialog Properties

Name Name of the Run configuration that contains this XML document. If multiple configurations exist, you can select the desired configuration using the dropdown. When you re-select the configuration the rest of the dialog is updated to reflect the settings from that configuration.

Click **Run** to save any modifications back to the configuration.

Local library Local library name. During XML document execution rules will be searched for in local, installation and system libraries. Use this field to specify the local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

XML Document Name of the XML document. You can use the **Browse...** button to select an XML document from the list of XML documents added to the project.

User Data Data to be used when you run the XML document. The data consists of Name – Value pairs that you specify using the **Add** button.

Update mode If selected, runs the XML document in update mode, which allows the XML document to perform updates to persistent database tables. If not selected, the XML document runs in browse mode, and any attempt by the XML document to perform an update to a persistent database table will cause the XML document to fail.

Note: In browse mode, EXP and VSM tables as well as memory-resident tables can be updated.

Using the Dialog

To use this dialog, perform the following:

1. In the **Local library** field, specify the name of the library (if not already populated).
2. In the **XML Document** field, specify the name of the XML document (if not already populated).
3. Specify user data if desired by selecting **Add** and typing the Name–Value pair into the table. Repeat as necessary, or select **Remove** to delete an existing Name–Value pair.

4. Select **Update mode** if desired.
5. Select **Run** to run; select **Cancel** to close the dialog without running.

Dictionary Field Selector Dialog

You use the Dictionary Field Selector dialog to select fields or parameters. This dialog displays under the following conditions:

- From the SUB Table editor, you select the **Add from Source** button to add a parameter or field to a table. For details, see SUB Table Editor on page 77.
- From the XML Field Maps editor, you select the **Add from table** button to specify a field to be mapped. For details, see XML Field Map Editor on page 128.
- From the EXP, IMP, MAP and VSM Table editors created when you use New Transaction Wizard. Select **Add from Cobol copybook** to add a field to a table.

The columns that display in the table vary depending upon which of the above actions was used to access this dialog.

Dialog Properties

OSB field name generation method

Select a method for generating OSB external field names from COBOL Copybook fields.

Using the Dialog

To use this dialog, perform the following:

1. Select one or more entries from the table. You select multiple entries by holding down the **Ctrl** key while clicking entries.
2. Select **OK** to acquire the values; select **Cancel** to close the dialog without making selections.

OSB Filter Dialog

You use the OSB Filter dialog to specify properties that are used to filter the objects that appear in a view.

Dialog Properties

Simple A set of fields, displayed in table format, for which you can specify values to filter objects. For each field, specify a pattern that can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character that must be present in the field.

When selected, the **Advanced** property is disabled.

Show system objects If selected, system objects are included when a filter is performed. Available only when **Simple** is selected.

Advanced Specify your own filter using fields of a particular TIBCO Object Service Broker metatable, for example For example, `NAME LIKE 'MY_TABLE*'` or `NAME='INVOICE'`. This is an advanced feature requiring knowledge of the metatables. For details, see Selection Criteria on page 155.

When selected, the **Simple** and **Show system objects** properties are disabled.

Using the Dialog

To use this dialog, perform the following:

1. Select **Simple** if you want to filter using one or more fields in the table.

   Specify values (in the **Filter** column) for the fields you want to include in the filter.

2. Select **Show system objects** if desired.
3. Select **Advanced** if you want to filter using your own criteria.

   Specify your criteria using the WHERE clause.

4. Select **OK** to run the filter; select **Cancel** to close the dialog without running the filter.

Log Filter Dialog

You use the Log Filter dialog to specify a date filter that restricts the transaction logs that appear in the Transaction Logs view. The filter works for a single date only; if you wish to see the transaction logs for multiple dates, you must run the filter for each date separately.

Using the Dialog

To use this dialog, perform the following:

1. Select a date from the list.
2. Select **OK** to run the filter; select **Cancel** to close the dialog without running the filter.

Table Data Selector Dialog

The Table Data Selector dialog lets you specify table parameter values for the Table Data editor and the Table Data browser. If the table has data parameters, you can select values from existing data instances of the table.

Dialog Properties

Parameters Fields for parameter entry. The parameters are specified in the table definition.

Existing Data Instances – opens the Data Instance Picker dialog that lets you select an existing data instance. Note that you can only browse data instances if you have DEF_PRM access to the table, or the table has an associated PRM table accessible to you.

Selection Criteria Optional selection criteria. For example, `FIELD1 LIKE "John*"` or `FIELD2="Invoice"`. For details, see Selection Criteria.

Local library Local library for the Table Data Editor or Browser session. See Local Library in Table Data Browser and Table Data Editor. You can use the **Browse...** button to select a library from the list of existing libraries.

Using the Dialog

To use this dialog, perform the following:

1. Enter the parameter values and/or press the Browse button and select a data instance.
2. Enter selection criteria or leave it blank to see all data in the table.
3. Specify the local library (if not already populated).
4. Select **OK**.

Selection Criteria

The selection criteria is used to select rows/items based on the following:

- The occurrences of a field
- The table instance of a parameterized table (optional method)

You can have from one to sixteen comparisons. These comparisons use relational and logical operators joined together by the AND (&) or OR (|) operators.

You can select the occurrences of a field that match a pattern using the pattern match operator LIKE. The pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character which must be present in the field.

Example:

```
REGION = 'MIDWEST' & ¬ (STATE_PROV = 'ONT' | STATE_PROV =
'MINN');
```

This example gets occurrences where the field `REGION` equals `MIDWEST` and the `STATE_PROV` field does not equal `ONT` or `MINN`.

Syntax

Note the following about notation:

- Square brackets indicate optional items.
- Curly brackets indicate items that can be repeated zero or more times. For example, curly brackets show that in a selection expression a selection term can be followed by zero or more items composed of a logical operator and a selection term:

  ```
  <SelExpr> ::= { <logicalOp> <SelTerm> }
  ```

- Round brackets indicate explicit grouping, in which case the round brackets must appear literally. For example, a negated selection expression that contains logical operators must be enclosed in parentheses to show that the whole expression, not just its first term, is negated:

  ```
  [NOT] ( <SelExpr> )
  ```

The syntax for the selection criteria is shown below:

```
[<Sel Predicate>]
<Sel Predicate> ::= <Sel NRelation>{<Logop><Sel NRelation>}
<Sel NRelation> ::= [<Not>} <Sel Relation>
<Sel Relation> ::= <Field Name> <Relop> <Expression>
<Sel Predicate>)

<Expression> ::= {<Unary op>}<Expr Term>
{<Addop><Expr Term>}
<Expr Term> ::= <Expr Factor>{<Multop>
<Expr Factor>}
<Expr Factor> ::= <Expr Primary>{**
<Expr Primary>}
<Expr Primary> ::= <Field Name>

<Constant>
(<Expression>)
<Logop> ::= <AND>
<OR>
<Relop> ::= =
=>
>=
<
<=
LIKE
<Addop> ::= +
-
||
```

DB2 Catalog Search Input Dialog

The DB2 Catalog Search Input dialog is used to specify search patterns for DB2 creators or schemas, for which DB2 table or stored procedure names should be loaded.

Dialog Properties

Creator/Schema DB2 creator pattern.

Table/Procedure DB2 table or stored procedure pattern.

Using the Dialog

To use this dialog, perform the following:

1. Enter a DB2 creator/schema and table/procedure patterns, or leave the fields blank to see all names.
2. Select **OK**.

Note: If the text entry fields are left blank, all DB2 table (stored procedure) names will be loaded without filtering.

DB2 Table/Procedure Selector Dialog

The DB2 Table/Procedure Selector dialog is used to select a DB2 table or a stored procedure.

Dialog Properties

Table with two columns: Creator/Schema and Name. DB2 table or stored procedure names are displayed in the table rows.

Using the Dialog

To use this dialog, perform the following:

1. Select the row with the desired DB2 table or stored procedure name.
2. Select **OK**.

Profiler Result Selection Dialog

The Profiler Result Selection dialog is used to select a profiler result from a list of previously executed rule profiling results.

Dialog Properties

List of profiler results with a filter input area to enter restrictions on the results displayed in the list.

Using the Dialog

To use this dialog, perform the following:

1. Enter the filter in the input area, or leave the field blank.
2. Select the profiler result from the list.
3. Select **OK**.

Chapter 7 Run and Debug Configurations

This chapter describes the Run and Debug configurations provided by the TIBCO Object Service Broker UI.

Topics

Rule Run and Debug Configuration Tabs

You use the Run and Debug configuration tabs to describe the name, arguments, and execution characteristics of a rule. Debugging does not require additional parameters, so the contents of Run and Debug configurations are identical.

Rule Properties

Name Name of the rule configuration.

Project Name of the project for the configuration.

Local library Local library name. To locate the rule, a search in local, installation and system libraries will be performed. Use this field to specify local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

The rule may no longer be present in the selected library. If not present, a rule with the same name but from the installation or system library will be run. If the search cannot locate the rule by name, an error occurs.

Rule Name of the rule. You can use the **Browse...** button to select a rule from the list of rules added to the project. Note that only rules from the library specified in **Local library** field are displayed.

Arguments Arguments used when you run the rule configuration. Arguments that have been defined for the rule appear in the **Name** column.

Refresh button Use this button to re-fetch the rule argument names. This is useful if the rule arguments have been modified.

Update mode If selected, runs the rule in update mode, which allows the rule to perform updates to persistent database tables. If not selected, the rule runs in browse mode, and any attempt by the rule to perform an update to a persistent database table will cause the rule to fail.

Note: In browse mode, EXP and VSM tables as well as memory-resident tables can be updated.

Test mode If selected, runs the rule in test mode. When a rule attempts to perform updates to persistent database tables, it is allowed to run successfully, but the updates are discarded.

Note: Most accesses to external databases are not affected by test mode, and changes to the databases are persistent. Also, in test mode, EXP and VSM tables are updated.

Profiling If selected, collects information about executing the rule (profile).

Data file name The back-end system file name to store profiling information.

To create, duplicate, or delete a rule configuration

Use the corresponding icon buttons in the top left corner. Note that the Run dialog creates a new configuration if an existing configuration is not available.

To run a rule configuration

1. In the **Name** field, specify the name of the rule configuration.
2. In the **Project** field, specify the name of the project for the configuration or use **Browse...** to find an existing project.
3. In the **Local library** field, specify the name of the library or use **Search** to find an existing library.
4. In the **Rule** field, specify the name of the rule or use **Browse...** to select a library from the list of existing libraries.
5. Specify the desired arguments.
6. Select **Update mode** if desired.
7. Select **Test mode** if desired.
8. Select **Profiling** to get a rule profile. Provide a data file name for Profiler.
9. Select **Apply** to apply changes to the rule configuration; select **Revert** to revert back to previous settings.
10. Select **Run** to run the rule configuration; select **Close** to close the configuration tab.

Transaction Run Configuration Tab

You use the Transaction Run Configuration tab to create, manage and run transaction configurations. Debugging of transactions is not supported, so corresponding Debug configurations are not available.

Transaction Properties

Name Name of the transaction configuration.

Project Name of the project for the configuration.

Local library Local library name. During transaction execution rules will be searched for in local, installation and system libraries. Use this field to specify the local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

Transaction Name of the transaction. You can use the **Browse...** button to select a transaction from the list of transactions added to the project.

User Data Data to be used when you run the transaction configuration. The data consists of Name – Value pairs that you specify or remove using the **Add** and **Remove** buttons.

To create, duplicate, or delete a transaction configuration

Use the corresponding icon buttons in the top left corner.

To run a transaction configuration

1. In the **Name** field, specify the name of the transaction configuration.
2. In the **Project** field, specify the name of the project for the configuration, or use **Browse...** to find an existing project.
3. In the **Local library** field, specify the name of the local library or use **Browse...** to select a library from the list of existing libraries.
4. In the **Transaction** field, specify the name of the transaction or use **Browse...** to find an existing transaction in the project.
5. Specify user data if desired by selecting **Add** and typing the Name–Value pair into the table. Repeat as necessary, or select **Remove** to delete an existing Name–Value pair.

6. Select **Apply** to apply changes to the transaction configuration; select **Revert** to revert back to previous settings.
7. Select **Run** to run the transaction configuration; select **Close** to close the configuration tab.

XML Document Run Configuration Tab

You use the XML Document Run Configuration tab to create, manage and run XML document configurations. Debugging of XML documents is not supported, so corresponding Debug configurations are not available.

XML Document Properties

Name Name of the XML Document configuration.

Project Name of the project for the configuration.

Local library Local library name. During XML document execution rules will be searched for in local, installation and system libraries. Use this field to specify local library to use during the search. You can use the **Browse...** button to select from the list of existing libraries.

XML Document Name of the XML Document.

User Data Data to be used when you run the XML Document configuration. The data consists of Name – Value pairs that you specify or remove using the **Add** and **Remove** buttons.

Update mode If selected, runs the XML Document configuration in update mode, which allows the XML Document to commit persistent updates to database tables. If not selected, the XML Document configuration runs in browse mode, and any attempt by the XML Document to commit a persistent update to a database table will cause the rule to fail.

Note: In browse mode, EXP and VSM tables as well as memory-resident tables can be updated.

To create, duplicate, or delete an XML document configuration

Use the corresponding icon buttons in the top left corner.

To run an XML document configuration

To use this dialog, perform the following:

1. In the **Name** field, specify the name of the XML Document configuration.
2. In the **Project** field, specify the name of the project for the configuration or use **Browse...** to find an existing project.

3. In the **Local library** field, specify the name of the local library or use **Browse...** to select a library from the list of existing libraries.
4. In the **XML Document** field, specify the name of the XML Document or use **Browse...** to find the desired XML document in the project.
5. Specify user data if desired by selecting **Add** and typing the Name–Value pair into the table. Repeat as necessary, or select **Remove** to delete an existing Name–Value pair.
6. Select **Update mode** if desired.
7. Select **Apply** to apply changes to the XML Document configuration; select **Revert** to revert back to previous settings.
8. Select **Run** to run the XML Document configuration; select **Close** to close the configuration tab.

Chapter 8 Debug Configurations

This chapter explains how the TIBCO Object Service Broker UI can be used to debug rules.

Topics

Debugging Rules

You can use the TIBCO Object Service Broker UI to debug rules. Breakpoints allow you to stop a rule execution and examine the current values of local variables and table occurrence buffers.

You can only have one run or debug session active at a time.

There are two ways to access the debugging feature:

- From the task menu, go to **Debug As > OSB Rule**.

  If multiple configurations exist, a selector is displayed. Choose the configuration you wish to debug. If no configurations are found, a new one is initiated and you are prompted for rule arguments and other settings.

- Select **Run > Debug Configurations** from the menu and create the Debug configuration manually.

Debugger Views

The TIBCO Object Service Broker UI uses standard debugging views of Eclipse. These include:

- **Debug** This view shows a call stack and buttons that resume or terminate the debugging session. When a breakpoint is hit these buttons become enabled. Note that you cannot suspend the execution.
- **Variables** This view displays current values of the variables and table fields from current occurrence buffers. Items modified since the last breakpoint are highlighted in yellow color. Note that the value modification is not supported.
- **Breakpoints** This is a list of defined OSB breakpoints. You can disable a breakpoint by unchecking the checkbox next to its name. The right-click menu in this view contains additional functions, including breakpoint deletion and properties viewing. See Using Breakpoints on page 172 for more information.
- **Console** This view displays textual messages relevant to the rule execution.

If any of the debug views are not visible, you can restore them:

1. Switch to the Debug perspective.
2. Right-click on the Debug button and select Reset.

Using Breakpoints

Add an OSB Breakpoint to stop a rule execution and examine the current values of local variables and table occurrence buffers. The rule executes until the breakpoint condition is met, then stops so that you can examine its current state.

Object Service Broker does not support stepping through a rule. Only breakpoints are supported.

Your Eclipse environment may switch to Debug perspective upon a break event.

OSB Breakpoints are added using the Run item in the main menu:

1. From the main menu, go to **Run > Add OSB Breakpoint...**
2. A breakpoint dialog appears. The dialog box lists break events. You can enter parameters pertinent to a specific event.

Note that breakpoints are not associated with a specific OSB project. Once a breakpoint is set, it remains in effect for all debug sessions.

Chapter 9 OSB Search

This chapter describes the search tabs and search views provided by the TIBCO Object Service Broker UI.

Topics

Object Search Tab

Use Object Search Tab to locate objects based on the specified criteria.

Fields

Criteria (this is a field group name)

Specify patterns to limit the set of objects that will be searched. A pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character which must be present in the field.

Field / Checkbox	Description
Name	Name pattern.
Unit	Unit pattern.
Summary/Description	Summary or description pattern.
Case sensitive	Check for case sensitive pattern matching of the summary/description.
System objects	If unchecked, system objects will be excluded.

Object Types

Select the object types you are searching for.

Checkbox / Button	Description
Rules	Check to include rules in the search
Tables	Check to include tables
Transactions	Check to include transactions
XML Documents	Check to include XML Documents
XML Field Maps	Check to include XML Field Maps
Screens	Check to include screens

Checkbox / Button	Description
Reports	Check to include reports
Select All	Use this button to check all the checkboxes
Deselect All	Use this button to uncheck all the checkboxes

Application

If Transactions is checked in **Object types**, enter the pattern for applications in which the transactions will be searched

Library

If Rules is checked specify in which libraries the rules will be searched

Checkbox	Description
Library name pattern	Check to include an additional library or libraries.
Library name pattern (text box)	Name pattern for the additional libraries to be included in the search. A pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character that must be present in the library name. You can use Browse... to specify an existing library.
Local library	Check to include the local library.
Installation library	Check to include the installation library.
System	Check to include the system library.

Using the Object Search Tab

1. Fill in the criteria.
2. Select object types.
3. Optionally fill in application and library patterns.
4. Select **Search** to run the search; select **Cancel** to close the dialog without running the search.
5. Access the results in the standard Eclipse Search view.

Rule Search Tab

Use Rule Search to locate rules that contain specified text.

Fields

Field / Checkbox	Description
Text	Text to find, must be specified.
Case sensitive	Check to make the search case sensitive.
Full text search	Check to enable full text search.

Rule Filter

Specify patterns to limit the set of rules that will be searched. A pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character which must be present in the field.

Field / Checkbox	Description
Name	Name pattern used to limit the rules that will be searched.
Unit	Unit pattern.
Summary	Summary pattern.
Case sensitive	Check to make the Summary pattern matching case-sensitive.
System objects	If unchecked, system objects will be excluded.

Library

Specify the libraries in which the search should be performed.

Checkbox	Description
Library name pattern	Check to include an additional library or libraries.

Checkbox	Description
Library name pattern (text box)	Name pattern for the additional libraries to be included in the search. A pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character that must be present in the library name. You can use Browse... to specify an existing library.
Local library	Check to include the local library.
Installation library	Check to include the installation library.
System	Check to include the system library.

Search Within Search Results

Specify if the search must be performed within a previous search results.

Field / Checkbox	Description
Search within search results	Check to search within the previous search results.
<previous search name>	Search result to search within.

Using the Rule Search Tab

1. Fill in text to search for.
2. Specify rule filter settings.
3. Select libraries.
4. Select **Search** to run the search; select **Cancel** to close the dialog without running the search.
5. Access the results in the standard Eclipse Search view.

Object Search View

The Object Search view displays the results of an OSB object search. The table consists of the following columns:

Column	Description
Name	Name of the object.
Type	The OSB object type.
Info	Object-specific information, such as library for a rule, application for a transaction, table type for a table.
Unit	Name for a group of related objects.
Summary/ Description	Explanation of the object.
Modified	Date the object was last modified.
Modifier	ID of the user who last modified the object.
Created	Date the object was created.
Author	ID of the user who created the object.

Working with OSB Objects

To perform a task on an object, locate the desired object in the table, then open the task menu. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired object.
- Right-click the text in the Name column of the desired

The task menu consists of the following selections:

Task Selection	Description	Applicable for
Open	Opens the selected object in the OSB object editor.	all
Add to Project…	Adds the selected object to the named OSB project. This option is valid for multiple selections.	all
Delete	Deletes the selected object. This option is valid for multiple selections.	all
Copy	Copies a reference to the selected object into the system clipboard. This option is valid for multiple selections.	all
Rename	Renames the selected object. When a prompt appears, type a new name for the object in the **New name** field.	all, but table
Show in Total	Shows a selected rule in the Profiler view Totals tab.	rule
Run As	Runs the selected object using the displayed Run Object dialog.	rule, transaction, XML document
Edit Data	Edit the data in the selected table using the displayed Table Data editor.	table
Browse Data	Browse the data in the selected table using the displayed Table Data browser.	table

You can also add objects to the Project view by dragging their names and dropping then in the Project view.

Rules Search View

The Rules Search view displays the results of a rule search. The table consists of the following columns:

Column	Description
Name	Name of the rule.
Type	The OSB object type.
Library	Library name.
Unit	Name for a group of related objects.
Summary	Explanation of the rule.
Modified	Date the rule was last modified.
Modifier	ID of the user who last modified the rule.
Created	Date the rule was created.
Author	ID of the user who created the rule.

Working with Rules

To perform a task on a rule, locate the desired rule in the table, then open the task menu. You can open the task menu using either of the following methods:

- Click the icon on the left of the Name column of the desired rule.
- Right-click the text in the Name column of the desired

The task menu consists of the following selections:

Task Selection	Description
Open	Opens the selected rule in the Rule editor.
Add to Project…	Adds the selected rule to the named OSB project. This option is only available if there is a project selected in the OSB Projects view. This option is valid for multiple selections.
Delete	Deletes the selected rule. This option is valid for multiple selections.
Copy	Copies a reference to the selected rule into the system clipboard. This option is valid for multiple selections.
Rename	Renames the selected rule. When a prompt appears, type a new name for the rule in the **New name** field.
Show in Total	Shows the selected rule in the Profiler view Totals tab.
Run As	Run the selected rule using the displayed Run Rule dialog.

You can also add rules to the Project view by dragging their names and dropping then in the Project view.

Chapter 10 Tips and Tricks

This chapter describes tips and tricks for using the TIBCO Object Service Broker UI.

Topics

Console View - Scroll Lock

Select the Scroll Lock command to change whether scroll lock should be enabled or disabled for all open consoles.

Creating New Objects Quickly

Using a New Wizard is not the only way to create a new TIBCO Object Service Broker object. When you use the Open task menu item in UI when the specified object is not found, a New Wizard is displayed. You can use that to speed up the development process.

Example 1

You are editing a rule and realize that you will need to call another new rule.

1. Type "CALL RULE2".
2. Right-click on RULE2 and select Open. A New Rule wizard displays with the name populated.
3. Press Finish and edit RULE2.

Example 2

You are creating a new transaction and would like to add an Initial rule which does not exist.

1. Type the name in the Initial rule field.
2. Right-click and select Open. This displays the New Rule Wizard.

Appendix A Rules Language Reference

This appendix provides selected reference information about the TIBCO Object Service Broker rules language. See *TIBCO Object Service Broker Programming in Rules* for complete information about the rules language and rule processing.

Topics

Lexical Elements

A rule is a sequence of lexical elements. Lexical elements are as follows:

- delimiters
- hexadecimal literals (zero or more characters inclosed between an X' and a single quotation mark)
- identifiers (including reserved words)
- numeric literals
- raw-data literals (zero or more characters enclosed between an R' and a single quotation mark)
- string literals (zero or more characters enclosed in single quotation marks)
- Unicode literals (zero or more characters enclosed between a U' and a single quotation mark)

See Also *TIBCO Object Service Broker Programming in Rules* for information about national character set support.

Delimiters

A delimiter is one of the following special characters:

```
| &  * ( ) - + = : ; ' , / < >
```

or one of the following compound symbols:

```
**  <=  >=  =  ||  R'  r'  U'  u'  X'  x'
```

Tokens

Each item that belongs to the group of lexical elements is called a token. For example, **, LOCAL, and '123 abc' are all tokens. Adjacent tokens can be separated by spaces, a delimiter, or a new line. An identifier or numeric literal must be separated in this way from an adjacent identifier or numeric literal. The only lexical elements that can contain a space are the string literal and the Unicode literal. String literals, hexadecimal literals, raw data literals, and Unicode literals may span more than one line; all other lexical elements must fit on a single line.

Hexadecimal literals are shown as X'xx...', raw data literals as R'xx...', and Unicode literals as U'xx...'. These tokens can be used only in stored selection strings in subview tables, in the Find and Select primary commands for shareable tools, and in the selection string of the COUNTOCCURRENCES and PARSE_TAM shareable tools.

In the examples above, items 123 and abc are two tokens when not enclosed in single quotation marks.

Reserved Words

The following names are reserved by the system as keywords in the rules language:

AND	ASCENDING	BROWSE
CALL	COMMIT	CONTINUE
DELETE	DESCENDING	DISPLAY
END	EXECUTE	FORALL
GET	IN	INSERT
LIKE	LOCAL	NOT
NULL	ON	OR
ORDERED	PRINT	REPLACE
RETURN	ROLLBACK	SCHEDULE
SIGNAL	TO	TRANSFERCALL
UNTIL	UPDATE	WHERE

CONTINUE, while a reserved word, is not currently used in the rules language.

Character Set

All rules language constructs are represented with a character set that is subdivided as follows:

Uppercase and lowercase letters

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Digits

```
0 1 2 3 4 5 6 7 8 9
```

Special characters

```
@ # $ |  & * ( ) - _ = + ; : ' , . / < >
```

the space character

If your system is NLS-enabled, these characters are mapped to the appropriate code page for your operating environment.

See Also *TIBCO Object Service Broker Programming in Rules* for information about national character set support.

TIBCO Object Service Broker National Language Support for information about NLS.

Operators

You can use the following types of operators with rules language expressions:

- Assignment Operator
- Concatenation Operator
- Arithmetic Operators
- Logical Operators
- Relational Operators

See Also Operator Precedence

Assignment Operator

The rules language uses the equal sign (=) as the assignment operator.

Simple Assignment of a Value

In simple assignment, a single value is assigned to a field of a table or to a local variable. Two examples of simple assignment statements are as follows:

```
CARS.PRICE = (PRICES.BASE + PRICES.SHIPPING) * TAXES.RETAIL;
AMOUNT = PRINCIPAL * (1 + INTEREST) ** YEARS;
```

Assigning Values by Name

In assignment-by-name, the field values of the table on the right are assigned to identically named fields of the table on the left. The field names are replaced with the asterisk symbol (*) for both the source and target tables. For example:

```
INPUTORDERS.* = ORDERS.* ;
```

Initializing All the Fields of a Table

Assignment-by-name is also a convenient way to initialize all the fields of a table. For example:

```
ORDERS.* = NULL;
```

This statement initializes all the fields in the ORDERS table to null values.

Concatenation Operator

The rules language uses a double vertical bar (||) as the concatenation operator. The concatenation operator is valid between any two semantic data types. If one of the operands has syntax W, the result has semantic type string and syntax W. In all other cases, the result always has semantic type string and syntax V.

Arithmetic Operators

The rules language contains the following operators for doing arithmetic:

Operator	Usage
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction
-	unary minus
+	unary plus

The arithmetic operators allow four types of operands: count, quantity, date, and typeless. The only operands allowed for unary - and unary + are count, quantity, and typeless.

Permitted Syntaxes for Arithmetic Operations

You can perform arithmetic operations with any of the syntax types, provided the semantic data type permits it. Arithmetic operations are performed on the following syntax types:

- Binary (B)
- Packed decimal (P)
- Floating point (F)

Converting Strings to Numbers

If an operand is of syntax C, UN, V, or W, the string is converted to a numeric syntax based on the value of the string. TIBCO Object Service Broker applies the following conversion rules to string data used in arithmetic operations:

- Binary is assigned for a string that is all digits.
- Packed decimal is assigned if the string contains a period (.).
- Floating point is assigned if the string has a numeric to the right and left of an exponent sign (E). An optional period (.) is also allowed before the exponent sign.

If a binary or packed field cannot hold the value, float is used.

If one operand of an arithmetic operation is of syntax RD, it is converted to the syntax of the other operand prior to the operation. Arithmetic operations are not permitted between two RD operands.

Converting Numbers to Strings

When a number is converted to a variable-length string, non-significant zeros are removed. For example:

The number...	becomes...
00010	10
040.0170	40.017
00500.0200E-013	500.02E-13

See Also *TIBCO Object Service Broker Programming in Rules* for information about arithmetic processing in TIBCO Object Service Broker.

Logical Operators

Logical operators join or negate expressions in rules language statements.

The following table lists the logical operators that you can use with the rules language (both the word and the symbol are valid). The logical operator OR (|) takes precedence over the logical operator AND (&).

Word	Symbol	Meaning
AND	&	The expression evaluates to true (Y) if and only if both logical operands are true. This operator can be used only in selection criteria.
OR	|	The expression evaluates to true (Y) if either logical operand is true, or if both are true. This operator can be used only in selection criteria.
NOT		The expression evaluates to true (Y) if and only if the logical operand is false (N). This operator can be used in the conditions part of a rule as well as in selection criteria. The only permissible syntax for use with LIKE is as follows: `WHERE NOT(fieldname LIKE 'value')` You can also enter the NOT operator by typing the caret (^) symbol.

Relational Operators

The following table shows the operators for making comparisons:

Operator	Description	Notes
=	equal to	
=	not equal to	On Windows and UNIX, this may display as ^=. You cannot substitute the keyword NOT for the not sign ( ) symbol.
<	less than	
<=	less than or equal to	

Operator	Description	Notes
>	greater than	
>=	greater than or equal to	
LIKE	pattern matching	Use only in selection criteria statements, not in condition statements. The only permissible syntax for use with NOT ( ) is as follows: `WHERE NOT(fieldname LIKE 'value')`

Semantic Data Type and Syntax Validations

Note the following points about semantic data types and syntax in expressions containing these operators:

- The relational operators for equality and inequality (=, =) can be used with any two operands of the same semantic data type. They can also be used for two operands of different semantic data types if one of the operands is typeless or if the types are identifier and string, or identifier and count.
- The relational operators for ordering (<, <=, >, >=) cannot be used with an operand of type logical. Otherwise, they can be used with two operands with the same semantic data type or if one operand is typeless.
- The relational operator for pattern matching (LIKE) can be used with any semantic data types and always returns a logical value.
- Trailing blanks are significant for variable length strings, but not for fixed length strings. For example, comparing two fixed length strings which have lengths 12 and 16 respectively has the same result as if the shorter string had been extended to length 16 and padded with four blanks on the right.
- If syntax differs, operands are converted to a common syntax. Refer to Arithmetic Operators for information on syntax conversions.
- The result of a comparison is always a logical value (Y or N).
- The LIKE relational operator is not case sensitive if you are matching syntax C data. In all other cases, the other relational operators distinguish between upper and lowercase.

See Also *TIBCO Object Service Broker Programming in Rules* for information about which semantic types can be used with the relational operators.

Operator Precedence

Operators within an expression conform to conventional notation and obey the precedence given below. Exponentiation has highest precedence and addition, subtraction, and string-concatenation have the same lowest precedence. In cases where more than one operator has the same precedence, such as addition and string-concatenation, the operators are evaluated strictly from left to right, unless explicitly overridden using parentheses.

Operator	Usage
**	exponentiation
*, /	multiplication, division
+, -	unary plus, unary minus
+, -, ||	addition, subtraction, string-concatenation

You must use parentheses if you intend to do two or more exponentiation operations consecutively. Two correct examples are as follows:

```
(A ** B) ** C
A ** (B ** C)
```

See Also *TIBCO Object Service Broker Programming in Rules* for information about arithmetic processing in TIBCO Object Service Broker.

Action Statements

The rules language has the following action statements:

- CALL
- COMMIT
- DELETE
- DISPLAY
- DISPLAY & TRANSFERCALL
- EXECUTE
- FORALL
- GET
- INSERT
- ORDERED
- PRINT
- REPLACE
- RETURN
- ROLLBACK
- SCHEDULE
- TRANSFERCALL
- WHERE

See Also *TIBCO Object Service Broker Programming in Rules* for information about action statement types.

CALL

The CALL statement invokes a rule within the scope of the current transaction. Upon completion, control passes to the next action in the calling rule. Arguments in CALL statements are passed by an argument list. All arguments must be specified when a rule is called.

Usage Notes

The called rule can be a rule in the installation library, system library, or the local library you will be using for your session when you execute the rule, or it can be a shareable tool.

You can invoke the rule directly or indirectly (using indirect referencing).

You cannot trap for the following: a non-existent rule, whether the rule name is not semantic data type I (for identifier) or typeless, or an incorrect number of arguments.

Exceptions

ROUTINEFAIL – Signaled if the called routine fails and the cause of the error cannot be signaled by other system exceptions.

RULEFAIL – Signaled if a table access resulting from the call is incorrect.

Examples

This example invokes a rule that contains one argument; the argument value is within the parentheses.

```
CALL EMPLOYEE_COUNT(10);
```

This example invokes the shareable tool ENDMSG which contains one argument, message. The value for message is a concatenation of the value for the local variable COUNT, the text string 'EMPLOYEES IN DEPARTMENT #', and the value for the argument DEPT passed in from a higher-level calling rule.

```
CALL ENDMSG(COUNT || ' EMPLOYEES IN DEPARTMENT #' || DEPT);
```

This example indirectly invokes a rule. In this example the value of the field ROUTINE, in the table FCNKEYS, is the name of the rule to be called.

```
CALL FCNKEYS.ROUTINE;
```

COMMIT

The COMMIT statement applies all changes made to TDS and external data since the last synchronization point. You do not need to explicitly issue a COMMIT at the end of a transaction. Data is implicitly committed for you at the end of the transaction.

Usage Notes

The FORALL statement only operates on data which has been committed to the database.

A GET statement that does not specify a primary key value only operates on committed data.

You can use the COMMIT statement before the commit limit is reached or when the exception ON COMMITLIMIT is signaled.

The COMMIT statement does not release locks. The locks are released on the affected table at transaction end.

When issuing a COMMIT, you can get an unusual and untrappable SYNC error resulting from the COMMIT updating too many page buffers in the Data Object Broker. The number of page buffers is determined by the WORKINGSET Data Object Broker parameter.

To avoid SYNC errors against updates to a parameterized table, issue a COMMIT at the end of the update(s) to each instance of the table.

Exceptions

COMMITLIMIT – Signaled if the maximum number of updates (INSERTs, REPLACEs, or DELETEs) between synchronization points has been reached.

Examples

In this example, the COMMIT statement commits the insertion to the EMPLOYEES table so that the inserted occurrence can be retrieved in the FORALL.

```
INSERT EMPLOYEES WHERE REGION = 'MIDWEST';
COMMIT;
FORALL EMPLOYEES WHERE REGION = 'MIDWEST':
```

In this example, a commit is made at the end of the updates to each instance of the EMPLOYEES table:

```
FORALL $EMPLOYEES:
  FORALL EMPLOYEES($EMPLOYEES.REGION):
  EMPLOYEES.SALARY = EMPLOYEES.SALARY + 100;
  REPLACE EMPLOYEES($EMPLOYEES.REGION);
  END;
  COMMIT;
END;
```

In this example, a COMMIT is issued when the exception COMMITLIMIT is signaled.

```
INSERT EMPLOYEES('MIDWEST');
_____________________________________
ON COMMITLIMIT:
COMMIT;
```

See Also

TIBCO Object Service Broker Parameters for information about Data Object Broker parameters.

DELETE

The DELETE statement removes an occurrence from a table in the database. This statement requires the primary key to be available in one of the following ways:

- Via a previous GET or FORALL on the table
- By explicit selection of the primary key
- By assigning a value to the primary key

Usage Notes

Field selection – You can specify which occurrence to remove by basing your selection criteria on the primary key field and using the equality relational operator (=). No other field selection is allowed.

Retrieving uncommitted data – If you delete an occurrence using a DELETE statement, you must commit the update to the table before you can retrieve the updated data. Otherwise, previously committed data which does not reflect your update is retrieved from the database. Refer to the COMMIT statement for more information about committing data.

Exceptions

DATAREFERENCE – Signaled if an error is detected in the specification of the selection criteria.

DELETEFAIL – Signaled if the occurrence does not exist in the table.

INTEGRITYFAIL – Signaled if an attempt is made to violate data integrity. Any of the exceptions lower in the hierarchy of the INTEGRITYFAIL group can also be signaled.

Examples

If a specific primary key value is selected using a WHERE clause, as shown in this example, the selected occurrence is deleted.

```
DELETE STUDENT WHERE ID = '810883';
```

If a GET or FORALL is performed without selection before the DELETE, as shown below, the first occurrence in the table is deleted.

```
GET STUDENT;
DELETE STUDENT;
```

DISPLAY

The DISPLAY statement shows a specified screen on a terminal or in a text window. You can access data that is entered on a screen using GET and FORALL statements on the screen tables of the screen. The DISPLAY statement can also be combined with the TRANSFERCALL statement and the UNTIL statement. Refer to the DISPLAY & TRANSFERCALL statement and the UNTIL … DISPLAY statement for more information.

Usage Notes To insert data from an application into a screen table, use the INSERT statement before the DISPLAY statement.

Exceptions DISPLAYFAIL – Signaled if an error was detected during an attempt to display a screen.

DEFINITIONFAIL – Signaled if the screen does not exist.

Examples In this example, data is retrieved from the MANAGERS table, assigned and then inserted into the screen fields of a screen table, and then the screen is displayed.

```
FORALL MANAGERS:
  MANAGER_DATA.* = MANAGERS.*;
  INSERT MANAGER_DATA('MANAGER_SCR');
  END;
DISPLAY MANAGER_SCR;
```

See Also *TIBCO Object Service Broker Defining Screens and Menus* for information about defining and presenting TIBCO Object Service Broker screens.

DISPLAY & TRANSFERCALL

In a text application environment, you can use the DISPLAY & TRANSFERCALL statement to improve concurrent access to the resources required by an application. You could use this statement under the following circumstances:

- You are creating an update application that will be used by more than one user at a time.
- You are accessing occurrences that may be required by other users.

Usage Notes The DISPLAY & TRANSFERCALL statement allows pseudo-conversational processing. This processing only controls resource utilization within TIBCO Object Service Broker. You can display a screen after ending the transaction, which frees the tables that the application uses but still gives you access to the information that is input to the screen.

Your screen environment is preserved across the transaction boundary. Anything associated with the screen, such as data occurrences, attributes set by screen tools, or cursor positioning, persists between displays unless you change the data or attributes.

You can specify whether the rule runs in browse or update mode.

Exceptions DISPLAYFAIL – Signaled if an error was detected during an attempt to display a screen.

DEFINITIONFAIL – Signaled if the screen does not exist.

Examples In this example, QUERY_SCREEN is the screen that is displayed and PROCESS_QUERY is the rule that can access input from QUERY_SCREEN. When a function key or <Enter> is used, PROCESS_QUERY begins as a new transaction.

```
DISPLAY QUERY_SCREEN & TRANSFERCALL PROCESS_QUERY;
```

See Also *TIBCO Object Service Broker Defining Screens and Menus* for information about defining and presenting TIBCO Object Service Broker screens.

EXECUTE

The EXECUTE statement starts a new transaction to invoke a rule. Upon completion, control passes to the next action in the original transaction. Like the CALL statement, the EXECUTE statement can invoke a rule directly or indirectly.

Usage Notes The executed rule can be a rule in the installation library, system library, or the local library you are using for your session when you execute it, or it can be a shareable tool.

You can invoke the rule directly or indirectly.

You can nest transactions using the EXECUTE statement.

You can specify whether the transaction runs in browse or update mode.

Exceptions EXECUTEFAIL – Signaled if an error was detected in the child transaction.

Examples This example directly invokes the rule EXECUTE EMPLOYEE_COUNT and passes in the argument value 10.

```
EXECUTE EMPLOYEE_COUNT(10);
```

This example indirectly invokes the rule whose name is the value of the field ROUTINE of the table FCNKEYS.

```
EXECUTE FCNKEYS.ROUTINE;
```

FORALL

The FORALL statement is a looping construct that processes a set of occurrences retrieved from the database. The body of the loop consists of the statements to be executed for each occurrence that satisfies the selection criteria. FORALL statements can be nested.

A FORALL statement contains the following:

- A table name, an optional WHERE clause, optional ORDERED clauses, an optional UNTIL clause, and actions.

- A colon (:), which follows the optional clauses (or the table name, if there are no clauses).
- Actions, which comprise the body of the loop and follow the colon. Each action starts on a separate line. Action sequence numbers are not permitted within a FORALL loop; since a FORALL loop constitutes a single statement, all actions within it are executed whenever a FORALL is executed.
- An END statement on a separate line which marks the end of the FORALL.

Usage Notes

If you make updates to a table, you must commit the updates to the database before you can retrieve them with a FORALL statement. Refer to the COMMIT statement for details about committing data.

You cannot issue a FORALL within a FORALL on the same table.

Selection within a FORALL

As with all table access statements, parameters and selection on fields are specified in a WHERE clause, as shown in the second example under Examples.

Ordering in Selection

When a FORALL statement is executed, table occurrences are selected in the order in which they are stored, unless a different order is specified by one of the following:

- One or more ORDERED clauses
- The ORD field in the table definition

The following shows an example of the ORDERED clause. In this example, the occurrences will be ordered by descending values of the field PRICE, then by ascending values of the field MODEL, and then by ascending values of the primary key LICENSE# (the default for ordering is ascending).

```
FORALL CARS WHERE CITY = INVOICE.CITY ORDERED DESCENDING PRICE &
ORDERED ASCENDING MODEL :
  CALL $PRINTLINE('CAR ID ' || CARS.LICENSE# || ' MODEL ' ||
CARS.MODEL || ' RETAIL PRICE:$' || CARS.PRICE);
END;
```

Exceptions

No exceptions are raised if occurrences are not selected by the FORALL statement. The actions in the body of the FORALL statement are not executed and processing continues for statements following the END statement.

A FORALL loop cannot contain a SIGNAL statement.

Any exception in the INTEGRITYFAIL group, except for COMMITLIMIT, can be signaled.

The DATAREFERENCE exception is signaled if an error is detected in the specification of the selection criteria.

Termination of a FORALL Statement

FORALL statement execution terminates under either of these circumstances:

- All occurrences satisfying the FORALL selection criteria have been processed.
- An exception is detected (and not handled by rules inside the FORALL loop) during the execution of the statements comprising the loop.

The table buffer is undefined after all occurrences satisfying the FORALL selection criteria have been processed. Accessing CARS.MODEL after the FORALL statement in Selection within a FORALL would not provide the model of the last car but would raise the UNASSIGNED exception.

Examples

This example retrieves all occurrences in the MANAGER table and calls PRINT_MANAGER to print the occurrences.

```
FORALL MANAGER :
  CALL PRINT_MANAGER;
END;
```

This example retrieves all occurrences in the MIDWEST table instance of the EMPLOYEES table for which the HIREDATE field has a value greater than the value of the BIRTHDATE field plus 40. The asterisk (*) represents the current table.

```
FORALL EMPLOYEES WHERE REGION = 'MIDWEST' & HIREDATE >
*.BIRTHDATE + 40 :
  ...
END;
```

This example retrieves all occurrences in the MANAGER table in which the MANAGER_NAME field ends in 'SON'.

```
FORALL MANAGER WHERE MANAGER_NAME LIKE '*SON' :
  ...
END;
```

To get all the manager names that do not end in 'SON', write the FORALL statement like this:

```
FORALL MANAGER WHERE  (MANAGER_NAME LIKE '*SON') :
```

This example retrieves all occurrences in the MANAGER table until the GETFAIL exception is signaled. Refer to UNTIL Statement for a description of how an UNTIL clause can be handled.

```
FORALL MANAGER UNTIL GETFAIL :
  ...
END;
```

See Also

TIBCO Object Service Broker Managing Data for information about TIBCO Object Service Broker tables.

GET

The GET statement retrieves the first occurrence in a table satisfying the specified selection criteria. You can specify the retrieval order of the occurrences by using the ORDERED clause.

Usage Notes

If you retrieve an occurrence that you have updated in the same transaction by specifying a unique primary key value in a GET statement, the uncommitted data is retrieved. Otherwise, previously committed data which does not reflect your update is retrieved from the database.

Exceptions

DATAREFERENCE – Signaled if an error is detected in the specification of the selection criteria.

GETFAIL – Signaled if there are no occurrences that meet the selection criteria.

INTEGRITYFAIL – Signaled if an attempt is made to violate data integrity. Any of the exceptions lower in the hierarchy of the INTEGRITYFAIL group can also be signaled.

Examples

This example retrieves the first occurrence in the STUDENTS table.

```
GET STUDENTS;
```

This example retrieves the first occurrence in the STUDENTS table, where the STUDENT# field has a value equal to 810883.

```
GET STUDENTS WHERE STUDENT# = '810883';
```

This example retrieves the first occurrence in the MONTHS table, where the MONTH field has a value equal to MM and the DAYS field has a value greater than or equal to DD. MM and DD are local variables which have been assigned a value in the parent rule.

```
GET MONTHS WHERE MONTH = MM & DAYS >= DD;
```

This example orders the occurrences in the MIDWEST instance of the EMPLOYEES table in descending order according to the values in the LNAME field, and then retrieves the first occurrence whose DEPTNO equals the value of the field DEPT of the INPUT table.

```
GET EMPLOYEES WHERE REGION='MIDWEST' & DEPTNO = INPUT.DEPT
ORDERED DESCENDING LNAME;
```

See Also

TIBCO Object Service Broker Managing Data for information about TIBCO Object Service Broker tables.

INSERT

The INSERT statement adds a new occurrence to a table. Occurrences within a table must have unique primary keys. No field selection is possible; the WHERE clause can only specify parameter values.

Exceptions

DATAREFERENCE – Signaled if an error is detected in the specification of the selection criteria.

INSERTFAIL – Signaled if an attempt is made to insert an occurrence with a primary key that already exists.

INTEGRITYFAIL – Signaled if an attempt is made to violate data integrity. Any of the exceptions lower in the hierarchy of the INTEGRITYFAIL group can also be signaled.

Examples

This example inserts data into the STUDENT table.

```
INSERT STUDENT;
```

This example inserts data into the parameterized CARS table. The values for the parameter CITY are provided by INPUT.CITY.

```
INSERT CARS WHERE CITY = INPUT.CITY;
```

This example inserts data into the EXPENSE_DATA screen table in the EMPLOYEE_EXPENSE screen.

```
INSERT EXPENSE_DATA WHERE SCREEN = 'EMPLOYEE_EXPENSE';
```

ORDERED

An ORDERED clause can be used in conjunction with a WHERE clause to retrieve occurrences in a specific order using the operators ASCENDING or DESCENDING.

By default, a table is ordered in ascending order by primary key, although this can be changed in the table definition.

Usage Notes

If an ASCENDING or DESCENDING operator is not specific, the order returned is ascending.

Multiple ORDERED clauses can be used in one access statement. Use the AND (&) operator to join the clauses.

If seven or more ORDERED clauses are used in a table access statement, the external sort program set up for your TIBCO Object Service Broker installation is used to sort the values.

Examples This example orders the occurrences of the MIDWEST instance of the EMPLOYEES table in descending order of LNAME field, then retrieves the first occurrence whose DEPTNO equals the value of the DEPT field of the INPUT table.

```
GET EMPLOYEES('MIDWEST') & DEPTNO = INPUT.DEPT ORDERED
DESCENDING LNAME;
```

This example retrieves the occurrences in descending values of the field PRICE, then by ascending values of the field MODEL, and then by ascending values of the primary key (the default for ordering is ascending).

```
FORALL CARS WHERE CITY = INVOICE.CITY ORDERED DESCENDING PRICE &
ORDERED ASCENDING MODEL :
```

See Also *TIBCO Object Service Broker Managing Data* for information about TIBCO Object Service Broker tables.

PRINT

The PRINT statement is used to send a report to a predetermined destination: the message log, printer or file. Your session options determine the destination; you can override the print destination options by a call to the $SETRPTMEDIUM tool. The PRINT statement can print a report directly by referring to the report name, indirectly by referring to a field of a table, or to an argument name that represents the report name.

Exceptions DEFINITIONFAIL – Signaled if the report does not exist.

Examples This example prints the report DEPT_SALARY directly.

```
PRINT DEPT_SALARY;
```

This example prints the report whose name is the value of the argument for a rule.

```
PRINT(REPORTNAME);
```

See Also *TIBCO Object Service Broker Defining Reports* for information about defining and producing reports.

TIBCO Object Service Broker Shareable Tools for information about shareable tools.

REPLACE

The REPLACE statement updates an occurrence in the database. You should first retrieve the data that you will modify using a GET or FORALL statement.

Usage Notes To alter the primary key value of an occurrence, you must DELETE the old occurrence and INSERT the new one.

No field selection is possible; the WHERE clause can only specify parameter values.

Exceptions

DATAREFERENCE – Signaled if an error is detected in the specification of the selection criteria.

INTEGRITYFAIL – Signaled if an attempt is made to violate data integrity. Any of the exceptions lower in the hierarchy of the INTEGRITYFAIL group can also be signaled.

REPLACEFAIL – Signaled if the primary key does not exist or if an attempt was being made to update a table from a rule running in browse mode.

Examples

This example replaces data in the STUDENTS table.

```
REPLACE STUDENTS;
```

This example replaces data in the parameterized table CARS; it is parameterized by CITY. The values for the CITY parameter are provided by INPUT.CITY.

```
REPLACE CARS(INPUT.CITY);
```

RETURN

The RETURN statement returns a specified result. The result can be an identifier or a constant. A rule that contains a RETURN statement is a function.

Examples

This example sets the return value to Y.

```
RETURN('Y');
```

This example returns the salary of the employee plus an increase.

```
RETURN(EMPLOYEE_SALARY.SALARY + INCREASE);
```

ROLLBACK

The ROLLBACK statement discards all changes made to TDS and external data since the last synchronization point. Locks are not released on the affected tables until the transaction ends.

Examples

In this example, a ROLLBACK statement is issued so that none of the changes made to the database since the last synchronization point are applied.

```
FORALL HIREDATE:
  GET EMPLOYEES('MIDWEST') WHERE EMPNO=HIREDATE.EMPNO;
  EMPLOYEES.HIREDATE = HIREDATE.DATE;
  REPLACE EMPLOYEES('MIDWEST');
  END;
-------------------------------------------------------
ON GETFAIL:
  ROLLBACK;
```

SCHEDULE

The SCHEDULE statement allows asynchronous processing to take place. The SCHEDULE statement makes use of an instance of the @SCHEDULEMODEL table to run a named rule using z/OS JCL, Windows batch programs, or UNIX scripts.

Usage Notes

You can specify whether the transaction runs in browse or update mode.

If you are using the z/OS version of TIBCO Object Service Broker, you can use the TO clause to schedule a rule to be sent to a queue for later processing.

By not including the TO clause, you can submit an instance of the @SCHEDULEMODEL table for immediate processing by the operating system.

For the Windows and UNIX platforms, the SCHEDULE statement automatically wraps, within single quotation marks, values that contain spaces or that have a length of zero. The substitution value for a zero-length string continues to be two single quotation marks.

Exceptions

LOCKFAIL – Signaled if a lock is held on the @SCHEDULEMODEL table (z/OS only).

SECURITYFAIL – Signaled if there is a security violation on the @SCHEDULEMODEL table (z/OS only).

Examples

This example submits the rule PRINT_INVOICE immediately for batch processing. The value for its one argument is provided by the INVOICE# field of the INPUT table.

```
SCHEDULE PRINT_INVOICE(INPUT.INVOICE#);
```

This example sends the CLEANUP rule to a queue called WEEKEND. This rule has one argument called LOCATION, and its value is provided by the CITY field of the INPUT table.

```
SCHEDULE TO 'WEEKEND' CLEANUP WHERE LOCATION = INPUT.CITY;
```

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for information about the use of queues.

TRANSFERCALL

The TRANSFERCALL statement terminates the current transaction and starts a new one. When the called rule is finished executing, the transaction is complete. If the transaction is called from a nested transaction, control passes to the next action in the parent transaction.

Usage Notes

You can specify whether the rule runs in browse or update mode.

The TRANSFERCALL statement can invoke a rule directly by referring to the rule name, or indirectly by referring to a field of a table, or to an argument name that represents the rule name.

Examples

This example invokes the rule SELECT_RENTAL directly.

```
TRANSFERCALL SELECT_RENTAL(NEAREST_CAR(LOCATION));
```

This example invokes the rule SELECT_RENTAL in browse mode.

```
TRANSFERCALL IN BROWSE SELECT_RENTAL WHERE RENTAL_LOC =
NEAREST_CAR(LOCATION);
```

This example invokes the rule whose name is the value of the field ROUTINE of the table FCNKEYS.

```
TRANSFERCALL FCNKEYS.ROUTINE;
```

WHERE

The WHERE clause is used to select the following:

- The occurrences of a field
- The table instance of a parameterized table (optional method)
- The arguments of a rule (optional method)

You can use it in conjunction with the ORDERED clause.

Usage Notes

You can select the occurrences of a field that match a pattern using the pattern match operator LIKE. The pattern can contain question marks (?) representing one character of any kind, asterisks (*) representing zero or more unspecified characters, and any other character which must be present in the field.

You can refer to the current occurrence of the current table with an asterisk (*).

You can have from one to sixteen comparisons in a WHERE clause. These comparisons use relational and logical operators joined together by the AND (&) or OR (|) operators. For details, see Operators.

Examples

This example gets the first occurrence from the EMPLOYEES table where the parameter equals MIDWEST and the STATE_PROV field does not equal ONT or MINN.

```
GET EMPLOYEES WHERE REGION = 'MIDWEST' &  (STATE_PROV = 'ONT' |
STATE_PROV = 'MINN');
```

This example shows how you can also specify selection criteria for a DELETE statement. The selection criteria must consist of an equality between a value and the primary key. In the example, MANAGER_NUM is the primary key for the MANAGER table.

```
DELETE MANAGER WHERE MANAGER_NUM = 80002;
```

This example shows how you can use the WHERE clause with any of the table access statements to select table instances. In the example, REGION is the parameter name and MIDWEST is the parameter value.

```
DELETE EMPLOYEES WHERE REGION = 'MIDWEST';
```

This example shows how you can use the asterisk (*) symbol to retrieve an occurrence that has the same value in two fields of the same table.

```
GET MANAGER WHERE MANAGER_NUM = *.USERID;
```

See Also *TIBCO Object Service Broker Managing Data* for information about TIBCO Object Service Broker tables.

Exception Statements

The rules language has the following exception statements:

- ON
- SIGNAL
- UNTIL
- UNTIL … DISPLAY

See Also *TIBCO Object Service Broker Programming in Rules* for information about exception handling in rules.

ON

The ON statement begins an exception handler. It includes the exception name and is coded in the exception handler portion of the rule. It may be followed by a sequence of actions that are executed if the exception is detected.

The ON statement consists of the following:

- The keyword ON
- The name of either a user-defined or system exception
- A colon (:)
- Actions, if coded. Each action starts on a separate line.

Usage Notes Action sequence numbers are not permitted within an ON statement or its following action statements.

Examples This example traps the exception GETFAIL on the MANAGER table. If a GETFAIL occurs on the MANAGER table, the rule ENTER_MANAGER is called. If a GETFAIL occurs on another table, an error message is issued.

```
ON GETFAIL MANAGER :
  CALL ENTER_MANAGER;
```

This example traps the exception DEFINITIONFAIL. If a DEFINITIONFAIL occurs, the shareable tool ENDMSG is called and a message is returned to the screen.

```
ON DEFINITIONFAIL :
  CALL ENDMSG('THE REPORT DOES NOT EXIST.');
```

SIGNAL

The SIGNAL statement raises the exception specified within the statement. You use the SIGNAL statement to raise user-defined exceptions within your rules.

A SIGNAL statement contains the following:

- The keyword SIGNAL
- The name of a user-defined exception
- A semicolon (;)

Usage Notes

An ON or UNTIL statement can detect an exception raised by the SIGNAL statement, and subsequently issue actions.

When using the ON or UNTIL statements, you must provide the exception name given in the SIGNAL statement; you cannot use an indirect reference in the SIGNAL statement. To use an indirect reference, use the $SIGNAL tool.

If you issue a SIGNAL statement from within a trigger or validation rule, the trigger or validation rule must explicitly handle the exception within its calling hierarchy. If it is not explicitly handled, the transaction which caused the trigger or validation terminates.

You cannot issue a SIGNAL statement within a FORALL statement.

Examples

This example signals the user-defined exception MISSING_INVOICE.

```
SIGNAL MISSING_INVOICE;
```

In this example, the GETFAIL exception signals the user-defined exception UNKNOWN_NAME if the GET fails on the MANAGER table.

```
ON GETFAIL MANAGER :
  SIGNAL UNKNOWN_NAME;
```

UNTIL

The UNTIL statement specifies an exception or a list of exceptions, each separated by the keyword OR. It allows looping to take place in rule execution. Looping terminates if an exception is detected. An END statement, on a separate line, marks the end of the UNTIL statement.

You can use the UNTIL statement on its own, or in conjunction with a FORALL statement and as part of the UNTIL … DISPLAY statement. For more information, see the FORALL statement and the UNTIL … DISPLAY statement.

An UNTIL statement contains the following:

- The keyword UNTIL
- An exception name, or a list of exceptions separated by the keyword OR

- A colon (:)
- The actions, which comprise the body of the loop. Each action starts on a separate line. Action sequence numbers are not permitted within an UNTIL loop; since an UNTIL loop constitutes a single statement, all of the actions within it are executed whenever the UNTIL is executed.
- An END statement, on a separate line

Usage Notes

If a loop terminates because of an exception, control passes to new actions as follows:

- If the exception is specified in an UNTIL statement for the loop, then the actions executed next are those following the END statement of the loop (control passes to those actions even if there is an ON statement for that exception in the exception handler part of the rule). Upon completion of those actions, the rule is finished executing and control passes to the caller.
- If the exception is not handled by the UNTIL statement for the loop but is handled by an ON statement in the exception handler part of the rule, the actions executed next are those listed in the ON statement. For details, see the ON statement.
- If the exception is not specified in an UNTIL statement for the loop or in an ON statement in the exception handler part of the rule, either the exception will be trapped by an exception handler in a rule higher in the calling hierarchy, or the transaction terminates with an error condition.

Examples

This example performs the operations within the UNTIL up to the point the user-defined exception NO_MORE_NUMBERS is issued.

```
UNTIL NO_MORE_NUMBERS :
  MULTIPLIER = MULTIPLIER + 1;
  CALL CHECKDIGIT(LENGTH($NUMBER));
  END;
```

This example performs the FORALL loop operations up to the point the system defined exception GETFAIL is issued.

```
FORALL SOURCETAB1 UNTIL GETFAIL :
  GET SOURCETAB2 WHERE LINE_NUM = SOURCETAB1.LINE_NUM;
  CALL COMPARELINES(SOURCETAB1.TEXT, SOURCETAB2.TEXT);
  END;
```

UNTIL … DISPLAY

The UNTIL … DISPLAY statement, which is a looping construct with a display, displays a screen repetitively until an exception is encountered.

An UNTIL … DISPLAY statement contains the following:

- The keyword UNTIL
- An exception name or a list of exceptions separated by the keyword OR
- A DISPLAY statement followed by a screen name
- A colon (:)
- The actions, which comprise the body of the loop, follow the colon. Each action starts on a separate line.

  Action sequence numbers are not permitted within an UNTIL loop; since an UNTIL loop constitutes a single statement, all of the actions within it are executed whenever the UNTIL is executed.
- An END statement, on a separate line

Usage Notes Execution of an UNTIL … DISPLAY statement terminates when an exception on the list is detected (and not handled by rules inside the UNTIL loop) during the execution of the statements comprising the loop. Control passes to the actions that follow the END statement of the loop.

Examples In this example, the QUERY_SCREEN screen is displayed up to the point the DONE exception is signaled during the display.

```
UNTIL DONE DISPLAY QUERY_SCREEN :
  CALL PROCESS_FCNKEYS('QUERY_SCREEN');
  END;
```

See Also *TIBCO Object Service Broker Defining Screens and Menus* for information about defining and presenting TIBCO Object Service Broker screens.

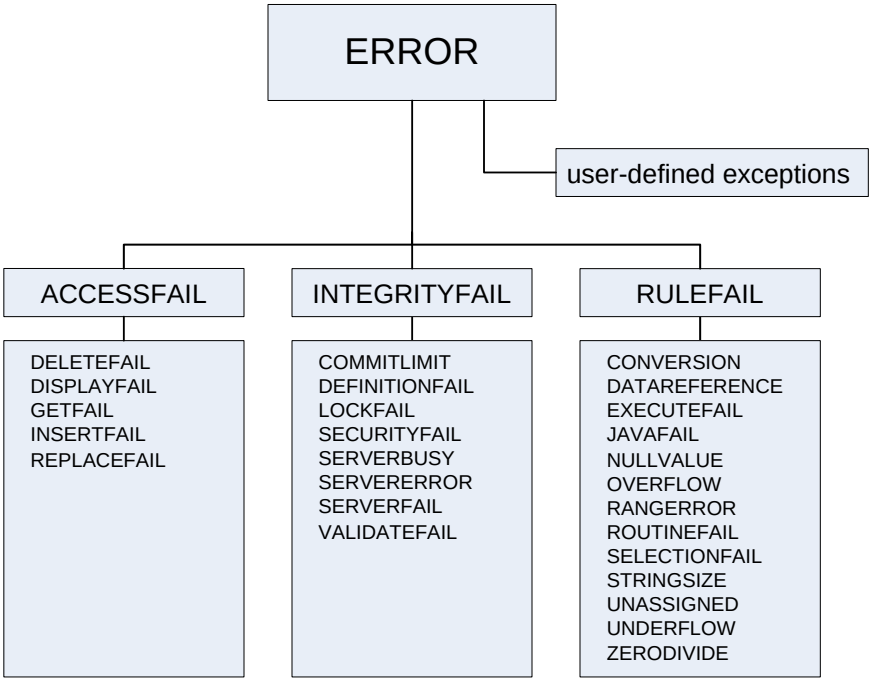
System Exceptions

This section describes system exceptions.

Hierarchy of System Exceptions

The TIBCO Object Service Broker run time environment signals system exceptions to permit an application to recover from an error. System exceptions form a hierarchy of names as shown in the following illustration.

Three levels of system exceptions are defined, and an exception traps any of the exception names that are below it in the hierarchy. The ERROR exception is at the highest level and traps all detectable errors that can be handled by an exception handler, whereas an exception such as GETFAIL only traps an error that occurs when a table occurrence is not found on the execution of a GET statement.



System Exceptions and Their Conditions

The conditions that trigger each of these exceptions are described below in alphabetical order. For more information, refer to *TIBCO Object Service Broker Programming in Rules*.

Exception	Condition
ACCESSFAIL	A table access error is detected or a rule running in browse mode is attempting to update a table.
COMMITLIMIT	The limit on the number of updates between synchronization points has been reached.
CONVERSION	A value has invalid syntax or cannot be converted to the target syntax.
DATAREFERENCE	An error is detected in selection criteria.
DEFINITIONFAIL	An error is detected in the definition of a table.
DELETEFAIL	The primary key for a DELETE statement does not exist or a rule running in browse mode is attempting to update a table.
DISPLAYFAIL	An error is detected when trying to display a screen.
ERROR	An error is detected.
EXECUTEFAIL	An error is detected in the child transaction.
GETFAIL	No occurrence satisfies the selection criteria.
INSERTFAIL	The primary key provided for an INSERT statement already exists or a rule running in browse mode is attempting to update a table.
INTEGRITYFAIL	Attempt to violate data integrity detected.
JAVAFAIL	A Java exception is raised by a called Java external routine.
LOCKFAIL	Another transaction is accessing this data in a way that prevents you from accessing the data.
NULLVALUE	An arithmetic operator is being applied to a numeric null or a numeric null is being passed as an argument value when a numeric value is required.

Exception	Condition
OVERFLOW	A value is too large to be assigned to the target syntax. The maximum value for the syntax is inserted into the receiving field. All tables are limited to the defined dictionary length. As well, screen and report tables are also limited to the display length.
RANGERROR	An argument to a given OSB routine is out of the allowable range.
REPLACEFAIL	A primary key provided for a REPLACE statement does not exist or a rule running in browse mode is attempting to update a table.
ROUTINEFAIL	A call to an OSB routine cannot complete successfully and a more specific system exception cannot be signaled.
RULEFAIL	An error results from incorrect rules coding, given that the dictionary definition of the database is correct.
SECURITYFAIL	Permission for the requested action is denied.
SELECTIONFAIL	An error is detected in a table occurrence during the evaluation of selection criteria.
SERVERBUSY	The requested external database server is not available to process the transaction.
SERVERBUSY	The requested external database server is not available to process the transaction.
SERVERERROR	External database server error detected.
SERVERFAIL	The connection to an external database server broke during a transaction or the external database server failed.
STRINGSIZE	The receiving string field is too short to contain the full length of the string value being assigned to it. The value is truncated to the length of the receiving field and inserted into that field.
UNASSIGNED	Reference is being made to a field of a table not assigned a value.
UNDERFLOW	A value is too small to be represented in the target syntax (usually exponent errors). The minimum value for the syntax is inserted into the receiving field.

Exception	Condition
VALIDATEFAIL	An attempt is being made to update a screen or table with invalid data. For example: • An attempt is being made to insert data into a table that failed a reference check or a non-Y value is being returned from a validation rule. • Invalid data is being inserted into a screen table from a rule (that is, the data failed the screen table reference check). • Invalid data existed on the screen when the user left the screen by using the Validation Exit key.
ZERODIVIDE	Attempt to divide by zero detected.

Index

P

R

S

T

V

W

X