TIBCO® Data Virtualization

User Guide

Version 7.0.7

Last Updated: *June 5, 2018*

Two-Second Advantage®

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 and the TIBCO logo are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries

TIBCO, Two-Second Advantage, TIBCO Spotfire, TIBCO ActiveSpaces, TIBCO Spotfire Developer, TIBCO EMS, TIBCO Spotfire Automation Services, TIBCO Enterprise Runtime for R, TIBCO Spotfire Server, TIBCO Spotfire Web Player, TIBCO Spotfire Statistics Services, S-PLUS, and TIBCO Spotfire S+ 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.

Copyright © 2002-2018 TIBCO Software Inc. All rights reserved.

TIBCO Software Inc. Confidential Information

Contents

Preface

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, please visit:

- https://docs.tibco.com

Product-Specific Documentation

The following documents form the TIBCO® Data Virtualization(TDV) documentation set:

- *TIBCO TDV and Business Directory 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.
- TDV Installation and Upgrade Guide
- TDV Administration Guide
- TDV Reference Guide
- TDV User Guide
- TDV Security Features Guide
- Business Directory Guide
- TDV Application Programming Interface Guide
- TDV Tutorial Guide
- TDV Extensibility Guide
- TDV Getting Started Guide
- TDV Client Interfaces Guide
- TDV Adapter Guide
- TDV Discovery Guide
- TDV Active Cluster Guide
- TDV Monitor Guide
- TDV Northbay Example

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website mainly in the HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Documentation for TIBCO Data Virtualization is available on https://docs.tibco.com/products/tibco-data-virtualization-server.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit https://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

Overview of Studio

This topic describes Studio, the main tool for working with TIBCO® Data Virtualization (TDV). It covers how to run Studio and describes its user interface.

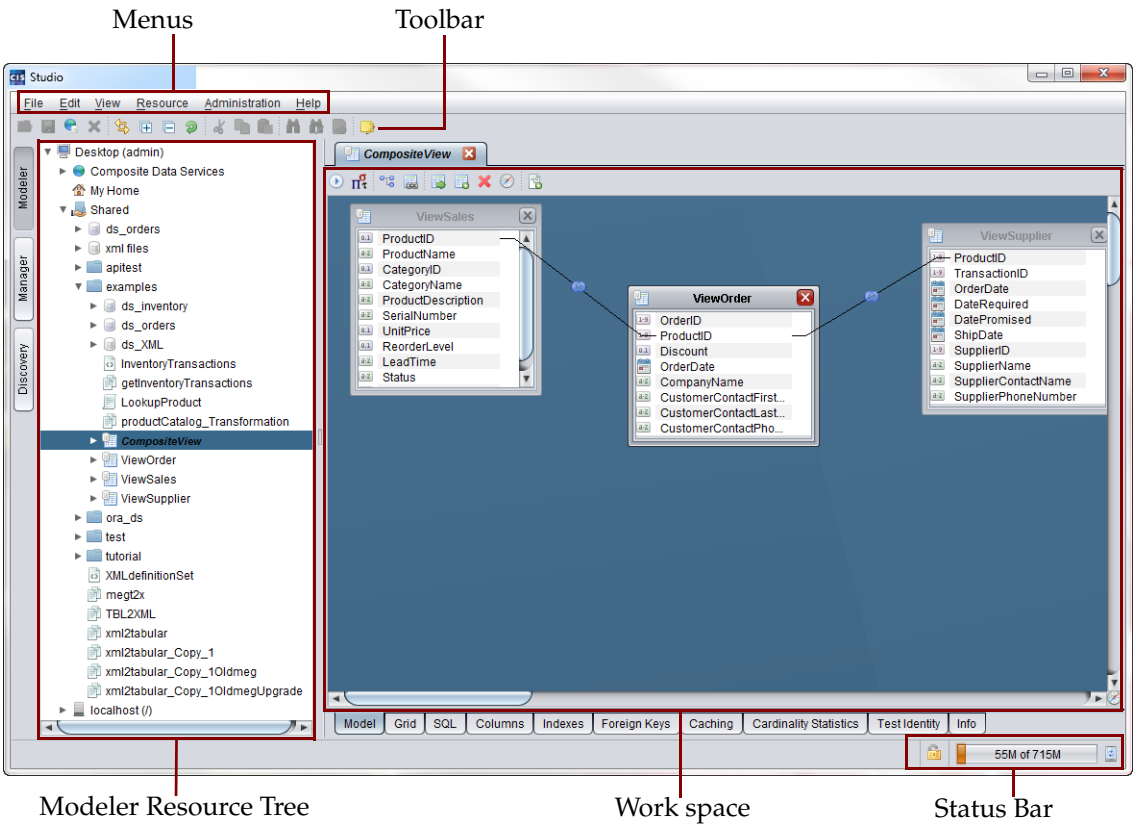
About Studio

Studio is the primary tool you use to work with TDV. Studio has three major components, accessed using tabs along the left side of the Studio window:

- Modeler
  - Model, manage, and publish data sources, transformations, views, SQL scripts, parameterized queries, packaged queries, definition sets, triggers, TDV databases, and Web services.
  - Define and refine resource properties and parameters using one of the work space editors.
  - Archive, export, and import TDV resources.
  - Configure TDV components and operational controls.
- Manager
  - Monitor and manage resource activities, including data source interactions with TDV. See the *TDV Administration Guide* for more information.

- Discovery
  - — Reveal hidden correlations in enterprise data so you can build data models for data virtualization and reporting.
  - — Scan data and metadata from across data repositories, whether they are applications, databases, or data warehouses.
  - — Using Discovery's graphical tools, create data models based on system metadata and discovered relationships. For details, see the *Discovery User Guide*.

The following graphic identifies the main components of the Modeler window.



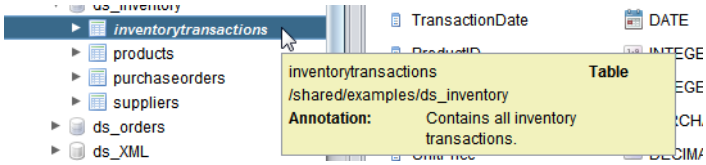
For more information about the Modeler resource tree, see Modeler Resource Tree, page 23.

The contents of the Modeler work space depends on the type of resource you have opened. Each resource has its own resource editor, and the editor usually has multiple tabs. To learn about the work space contents and how to work with each type of resource, refer to the appropriate topic of this manual.

Modeler Resource Tree

The Modeler resource tree is a hierarchical structure of all currently defined resources. TDV uses the term *resources* to collectively refer to all objects that are used for data modeling and building business solutions using TDV software. These resources include data sources, views, parameterized queries, SQL scripts, Java procedures, packaged queries, transformations, and TDV data services (which are available as TDV databases and Web services). See these sections for more details about the Modeler resource tree:

If you hover the pointer over the name of a resource, a tooltip displays information about the resource. The tooltip contents depends on the resource type.



The parent container path plus the resource name make up a unique identifier for invoking and referencing any TDV-defined resource. For example, the inventorytransactions table is referred to as /shared/examples/ds_inventory/inventorytransactions. This reference to this table is different from the XML schema definition set with the same name—/shared/examples/InventoryTransactions—both because the parent container path is different and because the name and path used to refer to the resource are case-sensitive.

Note: The resource tree displays only the resources the current user has permissions to view and use.

Location where Resource Tree Items Are Saved

The following table lists the relationship between the resources displayed in the resource tree.

Resource in the workspace and published areas	Resource in <TDV Host>
Desktop <current user>	/users/<domain>/<current user>

Resource in the workspace and published areas	Resource in <TDV Host>
Desktop <current user>/Data Services	/services
Desktop <current user>/Data Services/Databases	/services/databases
Desktop <current user>/Data Services/Web Services	/services/webservices
Desktop <current user>/My Home	/users/<domain>/<current user>
Desktop <current user>/Shared	/shared

The resource tree displays all system-created containers and all resources currently defined in TDV. When you create a new resource to use in the system, you add that resource to a container. None of the top-level system-created containers can be edited or deleted. The system-created nodes in the resource tree are described in the following sections:

- Desktop, page 24
- Data Services, page 24
- My Home, page 25
- Shared, page 26
- <TDV Host>, page 26

Desktop

The Desktop represents the current user's virtual work area on TDV Server. Some container names shown on the Desktop are shortcuts that provide easy access to containers deeper in the server's resource hierarchy (Data Services, My Home, Shared). The fourth container on the desktop, <TDV Host>:<port id>, represents the complete resource hierarchy of the server, including the contents of the other three Desktop containers.

Note: You cannot add a resource directly to the Desktop node of the resource tree.

Data Services

The Data Services folder contains two system-created containers, neither of which can be edited or deleted:

- Databases, into which you publish the resources that you want to make available to client applications that connect to TDV Server.
- Web Services, into which you publish the resources that you want to expose to Web Services clients.

Databases and Web Services within Data Services display all the resources published by any user in the system. Each published resource is tied to an unpublished resource residing elsewhere in the system.

For details on publishing, see Publishing Resources, page 407. For details on how clients can access the published resources, see the *TDV Client Interfaces Guide*.

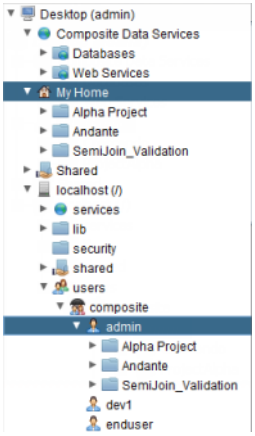
The **Databases/system** folder of the resource tree contains system tables that are used by the TDV system. For details on the contents of Databases/system, see the *TDV Reference Manual*.

Note: The system tables are subject to change with new releases of the system.

My Home

My Home is a semi-private (visible only to you and an administrator with full privileges) work area where you can design, develop, and test resources prior to exposing them as shared resource definitions or publishing them as externally available resources.

My Home is a shortcut to the directory folder <hostname>/users/<Domain>/<CurrentUser>. The name of the Studio user appears in parentheses at the top of the Studio navigation tree.



My Home cannot be deleted, and ownership of the container is ordinarily not reassigned. My Home can contain any other type of resource. These resources can reference other shared or published resources. For details about adding and publishing resources, see Publishing Resources to a Database Service, page 416.

Shared

Shared is a system-created container that contains all resources made available to all users in the system with appropriate access privileges. It is a place to store projects for teams to work on. For details about access privileges, see the *TDV Administration Guide*.

Shared/examples

This folder contains sample resources. For more information, see the *TDV Getting Started Guide*.

<TDV Host>

<TDV Host> is the name and port ID (in the form <TDV Host:Port ID>) of the TDV server to which Studio is connected. If Studio is connected to TDV on the local computer, the hostname is localhost by default. The default port number is 9400. <TDV Host> represents the entire contents of the TDV Server on your machine and has the following system-created containers:

Containers	Description
services	The services container has the same contents as the desktop folder named Data Services.
lib	The lib container contains all the system API calls. For details, see the *TDV Application Programming Interfaces Guide*.
security policy	This container holds security policies. TDV comes with a set of prebuilt security policies and if you define custom security policies, they are saved in this location.
shared	The shared container has the same contents as the desktop folder named Shared. When you add any resource to <TDV Host>/shared, the change is reflected in the structure of Desktop/Shared. For details, see Location where Resource Tree Items Are Saved, page 23.
policy	Contains security and cache policies.

Containers	Description
users	The users container has one container for each security domain in the server. The default domain composite is system-created, and cannot be edited or deleted. The system-created user named admin belongs to the domain composite. Each domain in users is represented by a folder-container, and each domain has a container for each user in that domain. These containers are referred to as those users' "home folders." You cannot add a resource directly to the users node, but when an administrator adds a user to a domain in the system, the new user's name and resources are displayed here. For details on adding users to the system, see the *TDV Administration Guide*. From users, you can view other users in the system and use their resources if you have appropriate access privileges. If you view your home folder from users, you see your Desktop/My Home, because My Home represents your home folder.

Opening Studio and Switching User

Starting Studio

Studio provides access to TDV and its resources. When you start Studio, you need to know the TDV Server instance to which you want to connect, the domain, and the port ID. For details, see the *TDV Installation and Upgrade Guide* or the *TDV Getting Started Guide*.

To start Studio

1. Select Start > All Programs > TIBCO > Studio <ver> > Start Studio <ver> from the Windows desktop. This command opens the Studio sign-in window.
2. Enter your user credentials and information for the TDV to which you are connecting. For details, see the *TDV Installation and Upgrade Guide* or the *TDV Getting Started Guide*.
3. Click Connect.

4. Click OK.

Studio opens on the Modeler tab (top tab along the left edge of the window.

See About Studio, page 21 for information about how the Studio user interface is organized.

Starting Studio from the Command Line

Studio provides access to the TDV Server and its resources. When you start Studio, you need to know the TDV Server instance to which you want to connect, the domain, and the port ID. For details, see the *TDV Installation and Upgrade Guide* or the *TDV Getting Started Guide*.

To start Studio from the command line

1. Open a UNIX or Windows command line tool.
2. Type one of the following:

Command	Description
studio.bat	Starts Studio from a DOS prompt.
studio.sh	Starts Studio from a UNIX prompt.
studio.bat -server=hostname	Starts Studio from a DOS prompt for a specific TDV Server.
studio.sh -server=hostname	Starts Studio from a DOS prompt for a specific TDV Server.

3. Enter your user credentials and information for the TDV to which you are connecting. For details, see the *TDV Installation and Upgrade Guide* or the *TDV Getting Started Guide*.
4. Click Connect.
5. Click OK.

Studio opens on the Modeler tab (top tab along the left edge of the window.

See About Studio, page 21 for information about how the Studio user interface is organized.

Changing the Connection Type

You can change users, domains, TDV Servers, ports, or connection security level. The TDV system administrator must set the Java Key Store for use in securing the SSL connection. For details, see the *TDV Administration Guide*.

To change the user, port, domain, or connection type, and then restart Studio

1. On the File menu, choose Switch User.
2. In the dialog box that appears, confirm that you want to switch user by clicking Yes.
3. Supply your username and password, and enter the domain, server, and port information.
4. (optional) If the connection must be secured, check the Encrypt check box to initiate an SSL connection between Studio and TDV.
5. Click Connect. Studio reopens.

Customizing Studio

This section contains topics that you can use to customize your Studio experience.

- Customizing Studio Using the Studio Options Dialog, page 29
- Customizing Studio Result Panel XML Text, page 35

Customizing Studio Using the Studio Options Dialog

The Studio Options dialog lets you configure Studio behavior locally. You can customize:

- What information is displayed in the title bar.
- What toolbars are displayed.
- Whether version control is enabled.
- Whether logging is enabled.
- How warnings and confirmations are handled.
- How editors are displayed in the workspace.

Other Studio instances installed on different computers that log onto the same TDV Server are not affected by these local Studio options.

To customize Studio

1. Open Studio and log in.
2. Select Edit > Options.

Studio opens the Options dialog box with the General tab displayed.

3. Edit the options as necessary.

Check box	Description
Display version, server host name, and port number in the title bar	Useful when the Studio user can connect with any one of several instances of TDV.
Display custom name in the title bar	Enter a string to display your text in the Studio application title bar.
Store execution parameters on disk and reuse when Studio runs again	Makes this Studio instance remember the procedure input parameters used during Studio test execution, so the test is easier to duplicate.
Copy privileges from parent folder when creating new resources	Sets the default privileges to apply creating new resources, making child resources accessible to the same groups and users.
Use a colored background in the View Model	Toggles the View Model background between blue and white.
Show the toolbar at top of main window	Toggles display of the Studio toolbar.
Show the status bar at bottom of the main window	Toggles display of the Studio status bar. This bar shows encryption status of the connection between the Studio and TDV, Studio memory usage, and introspection progress (when underway).
Show the navigation tree in main window	Toggles display of the Studio resource tree in Studio Modeler. This setting also toggles display of the Manager navigation panel.
Enable VCS	Select this option to enable PDTool features. PDTool is available as an open source tool through GitHub and professional services can be contacted to help you with your implementation.
Logging - Enable additional logging within Studio	Adds logged detail to assist with debugging. This can add a lot of information to the log file but has little affect on performance. Additional logging information is sent to the <TDV_install_dir>/logs/sc_studio.log file, which you can view using the View Studio Log button.

Check box	Description
Data Source Introspection - Cache Loading Page Size	This value adjusts the page size to use for cache during introspection.
Session Heartbeat(millisecond)-Session Heartbeat Interval	This value adjusts the Studio session timeout. After a timeout, Studio prompts you to sign in again.

4. Select the Warnings tab and edit the options as necessary.

   Warnings and confirmations check boxes can reset display of notifications and confirmation dialogs. These settings can be indirectly unchecked by dismissing a notification confirmation dialog after checking "Do not show me this warning again."

Check Box	Description
Mark folders impacted when they contain descendants that are impacted.	Selected, by default, to show indications that folders contain resources that are impacted by changes or have error that might need correcting. Clear to remove the warning icon from Studio resource tree folder objects. The warning will continue to appear next to the impacted resource regardless of the selection made here.
Warn before execution when other editors are unsaved	When checked, warns that a procedure will return results based on the resource metadata definitions saved on the server, and not as-yet unsaved definitions in editors currently open on the Studio desktop.
Warn when canceling the execution of a query.	
Warn before data lineage calculation when current editor is unsaved	When checked, warns that the data lineage calculation will return results based on resource metadata definitions on the server, rather than metadata changed but not yet saved in an open editor.
Warn when an OLAP View member is selected on Where tab, but its dimension is not selected on Select tab	
Warn when OLAP View members from same dimension but different hierarchies are selected on Where tab	

Check Box	Description
Display confirmation prior to closing Studio.	Display Confirmation check boxes display warning prompts even if warnings have been marked as not to be displayed again.
Display confirmation prior to clearing all relationship discovery tasks.	Displays confirmation prior to removal.
Display confirmation prior to removing the view model.	
Display confirmation prior to removing the OLAP view model.	

5. Select and edit the options on the Editors tab as necessary.

Some Studio text editor formatting and font display options are configurable. Each of the groups has a Reset button on the right, and the Editors tab has an overall Restore Defaults button at the bottom.

Option	Description
Automatic Capitalization	Capitalizes and formats SQL keywords when they are typed or pasted. You can set font, color and size using the Studio Options dialog.
Automatic Indentation	Automatically indents text based on the SQL keyword on the previous line. The following keywords have the same indentation, and the SQL Editor indent the line that follows a keyword in this list: `SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, FULL OUTER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, INNER JOIN, UNION, UNION ALL, EXCEPT, EXCEPT ALL, INTERSECT,` and `INTERSECT ALL`. Additionally newlines made with an Enter keystroke start with the indentation present on the preceding line, whether indentation is done with spaces or tabs. A single Backspace keystroke puts the cursor back at the far left position of the line.

Option	Description
Indent with Spaces (not tabs)	Allows insertion of spaces (ASCII character 32) for indentation instead of tabs (ASCII character 9). Both are handled as required for querying the underlying data sources. By default, tabs are used for indentation. Tab width lets you change to the default tab width as represented by spaces. Manually entered tabs can differ from the default SQL display of four spaces used as indentation.
Tab Width	Lets you change tab spacing from four (default) character widths to any value between 1 and 32.
Reset Format Defaults	Restores formatting options to their default values.
Comment, Keyword, Literal, Label, Operator, Invalid	Offers you a palette of colors to apply to categories of text in SQL statements. You can select each color from a group of swatches, from a hue-saturation-brightness (HSB) model, or (using sliders) from a red-green-blue (RGB) model.
Reset Color Defaults	Restores formatting options to their default values.
Font list	Lets you select a font family from the scroll list.
Font size list	Lets you select the font size from the scroll lists.
Bold	Check box that lets you select or unselect bold font.
Italic	Check box that lets you select or unselect italic font.
Reset Font Defaults	Restores font options to their default values.
scrollable field	Lets you preview what the SQL text looks like with your font family, size, and bold or italic choices.

6. Select and edit the options on the SQL Scratchpad tab as necessary.

   Some Studio SQL Scratchpad editor options are configurable in the SQL Scratchpad tab of the Studio Options panel. See Views and Table Resources, page 215 for more information about how to use the SQL Scratchpad. Each

group has a Reset Settings button on the right, and the SQL Scratchpad tab has an overall Restore Defaults button.

Item	Description
Open SQL Scratchpad Editor when Studio starts if it was open at exit check box	When checked, causes the SQL Scratchpad editor to open automatically when Studio restarts in cases where it was open when Studio was last exited. Default: checked.
Automatically save content when the editor is deactivated check box	When checked, causes Studio to save the SQL query currently being displayed in the SQL Scratchpad editor, as well as all queries in the History list, when you close the SQL Scratchpad editor. Default: checked.
Maximum History Size drop-down list	Lets you retain between 2 and 50 of the most recent SQL queries you have created in the SQL Scratchpad editor. Locked queries in the History list do not count toward this maximum. Default: 12.

7. Select and edit the options on the Transformation Editor tab as necessary.

Item	Description
Maximum Operation Width	Set the maximum width of the operations shown on the Transformation Editor model page.
Error Level to Show in the Editor	Defining a transform can be complicated, you can configure the level of error reporting displayed by Studio while you are defining your transform.
Show Insert Cast Dialog	When adding a cast function to your transform, you are shown a dialog to help you define the details of the cast. Prefixes are also used when defining namespaces. Seeing the namespace can help you avoid namespace conflicts.
Show Prefixes in Type Names	The prefix values are used to indicate function type. They can be used to identify if a function is SQL or custom.
Show Operation Annotations	For the operations shown on the Transformation Editor, display any annotations in the heading portion of the operations.

8. Save your changes.

Customizing Studio Result Panel XML Text

Many of the resources that you execute within Studio produce XML result sets that can be viewed in the Studio Result panel. Occasionally the XML displayed in the Result panel does not format well. If your XML isn't formatting well in the Result panel, it means that you are not retrieving the full result set. Studio has a configuration parameter that can be used to manipulate the total number of characters retrieved in XML results.

To customize the Result panel XML text

1. Validate that your XML for a resource is not formatting well in the Result panel. For example, lines are run together.
2. Select Administration > Configuration.
3. In the Configuration window, navigate to Studio > Data > XML Text Size.
4. Increase the value of the total number of characters to retrieve in the XML result.
5. Click OK.
6. Execute the resource that previously had XML formatting problems.
7. Validate that the XML file is formatting correctly.

If your XML is still not formatting, continue to increase the value of the XML Text Size parameter until the XML formats as expected.

Modifying Studio Memory Usage

You can change the extended memory and maximum memory allocation pool values for Studio.

To change your Studio memory settings

1. Select Edit > Options.
2. On the Memory tab, type values for the following:

Field label	Description
Xms	The initial memory allocation pool or eXtended Memory Specification for a Java Virtual Machine (JVM).
Xmx	The maximum memory allocation pool for a Java Virtual Machine (JVM).

3. Optionally, choose to have Studio restart.
4. Optionally, choose to have Studio stop showing memory is low messages.
5. Click OK.
6. Make sure that Studio restarts and that your new settings are applied.

Changing Your TDV Password

You can change your password using Studio's File menu, if you have the Access Tools right.

Changes made to the user rights profile take effect nearly immediately because TDV checks for appropriate rights every time feature access is attempted.

To change your own password using Studio

1. Select File > Change Password.
2. In the Change Password window:
   a. Type the old password in the Old Password field.
   b. Type the new password in the New Password field.
   c. Type the new password again in the Confirm Password field.
3. Click OK.

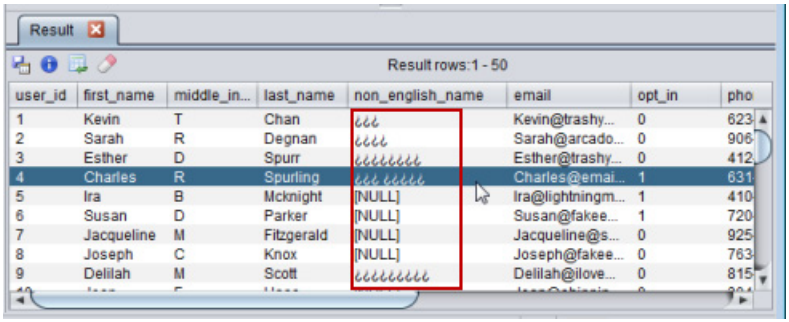
Note: "******NOT_KNOWN******" is a reserved token string. Do not use it in any TDV password fields.

Studio Unicode Support

The TDV Server returns Unicode characters in all messages carrying data. The TDV Server transforms messages to Unicode. Studio can be configured to display all the Unicode TrueType fonts to display those characters:

- Upgrading Studio to Display Unicode, page 37.

Here is a Studio result set showing Unicode displayed as upside-down question marks.



Upgrading Studio to Display Unicode

You can upgrade Studio to display data using Unicode TrueType fonts when it is present in column names or returned results.

To upgrade Studio to display Unicode

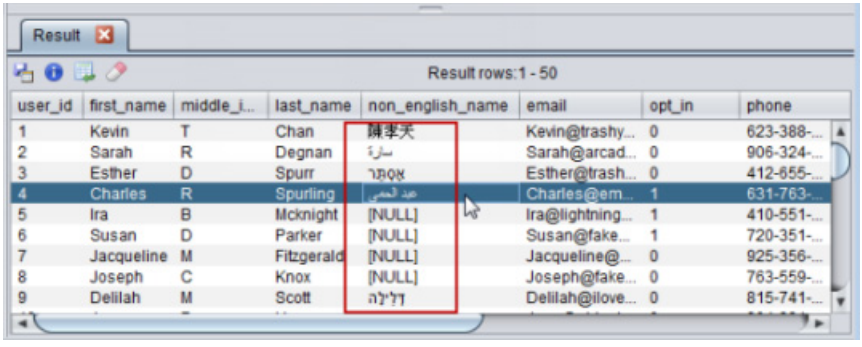
1. On the computer where Studio is installed, create a folder named fallback in the following path:

   <TDV_install_dir>jre\lib\fonts\fallback

2. From the directory C:\Windows\fonts or C:\WINNT\fonts, copy the Arial Unicode MS (TrueType) file named ARIALUNI.TTF, and paste it into the fallback directory.

3. If Studio is running, restart it.

   For example, with TrueType enabled the Results panel displays non-standard characters.



Security Features

Security features are discussed throughout this guide, but especially in these topics and sections:

- HTTPS

  Publishing resources (Publishing Resources, page 407)

- Kerberos

  Configuring and connecting data sources (Configuring Relational Data Sources, page 77, Configuring Web-Based Data Sources, page 105,Configuring File Data Sources, page 165)

  Publishing resources (Publishing Resources, page 407)

- Passwords

  In studio (Changing Your TDV Password, page 36)

  Working with data sources (Working with Data Sources, page 65)

  Working with views (Views and Table Resources, page 215)

  Publishing resources (Publishing Resources, page 407)

  Setting up and using caching (TDV Caching, page 465)

  Setting up and using data ship (Data Ship Performance Optimization, page 597)

- SSL

  In Studio (Customizing Studio, page 29)

  Working with data sources (Working with Data Sources, page 65)

  Publishing to a database service (Publishing Resources to a Database Service, page 416)

Studio Resource Management Overview

This topic describes how to create and work with TDV resources in Studio.

About Resource Management

Resource management involves creating and manipulating containers and resources in the TDV system. TDV resources are defined as the objects that are created, stored, and accessible in Studio including data sources, views, procedures, models, and the containers in which resources are organized. The containers are often referred to as folders.

The management of TDV resources includes such actions as creating, editing, copying, exporting, deleting, renaming, and moving the resources in the Studio resource tree.

Consider the following:

- To make a resource available to client programs, you must publish that resource. See Publishing Resources, page 407.
- APIs are available (in /services/webservices/system/admin/resource/operations) for resource management. See the *TDV Application Programmer Interface Guide* for details.

- Built-in procedures are available in the TDV system library (in /lib/resource) for managing resources. Refer to the *TDV Reference Guide* and the *TDV Application Programmer Interface Guide* for details.

Creating a TDV Resource

When you create a resource, you are adding an instance of the resource to the metadata repository. For a description of the hierarchy of user-created resources, refer to Modeler Resource Tree, page 23.

The process you follow and the resource access rights depend on the type of resource, as described in these sections:

- Locating a Container for a TDV Resource, page 40
- Creating a Resource, page 41
- Copying Privileges to Another Resource, page 41

You must grant access and use privileges explicitly to other users and groups before they can use the resource you create. For details on access privileges, see the *TDV Administration Guide*.

Locating a Container for a TDV Resource

You can create a TDV resource in your home folder (My Home), the Shared node, or in someone else's home folder, as long as you have the Write privilege on that container.

To find a container for a TDV resource

1. Right-click the container to open the context menu.
2. If New <resource_type> is listed for the resource-type in the menu that you want, you can create the resource in that container. Select that item and follow the procedure in Creating a Resource, page 41.

Or

1. Select a container.
2. Select File > New. If <resource_type> is listed in the submenu, you can create the resource in that container. Select that resource type and follow the procedure in Creating a Resource, page 41.

Creating a Resource

This section describes the common procedure for creating TDV resources. Several basic resource types are created in the same way. These resource types include:

- Definition sets (SQL, XML, or WSDL)
- Folders
- Models
- Parameterized queries
- SQL scripts
- Triggers
- Views
- XQuery procedures

Creation and configuration of other resource types are described elsewhere:

- Data sources are discussed in Working with Data Sources, page 65 through Configuring File Data Sources, page 165.
- Packaged queries are discussed in Procedures, page 267.
- Transformations are discussed in Using Transformations, page 311.

To create a basic TDV resource

1. Right-click the container where you want to add the resource, and select New <resource_type>; or select the container and then File > New > <resource_type>. Locating a Container for a TDV Resource, page 40
2. Type a name for the resource in the input field, and click OK.
3. To apply the privilege settings of the selected parent folder to the new resource object, check the Copy privileges from parent folder check box.

   By default, only you and administrators with Modify and Read All Resources rights can use the resource you create. If you do not elect to copy privileges from the parent folder, you need to modify resource privileges directly to let others share or use that resource.
4. Click **OK**.

Copying Privileges to Another Resource

You can set privileges on a resource in several ways. One way, described here, is to copy privileges from an existing container or resource to another container or resource.

To model privilege settings on the settings of an existing resource

1. Right-click select the container or resource that has the privileges you want to apply.
2. Select the Copy Privileges function. A window for selecting the target resource opens.
3. Select the container or resource that is to receive the copied privileges.

   Note: The target container or resource receives new settings only for privileges common to the source and target. For example, privileges for a table are a subset of privileges for a container, so only table-level privileges are changed.

   If you want all the resources contained in the selected target to have the same privileges, check the Copy Privileges into Target Descendants check box.

If you do not copy permissions from the parent folder into the target descendants, you or the administrator must modify the descendants' privileges directly.

Opening a Resource

To open a resource

- Double-click the resource name in the resource tree.
- Right-click and select Open from the popup menu.
- Select the resource and from the File menu, choose Open <resource name>.
- Select the resource and type Ctrl-o.

Getting Information about a Resource

You can get information about a resource like who owns it, who created it, and when it was created as well as any annotations about the resource that might have been added.

The data source tabs display the following information about the resource.

Field	Description
Name	User-provided name and location.
Type	Resource type.

Field	Description
Owner	Name of the owner.
Orig Creation Date	Date the resource was created. If created in a release prior to TDV 6.2, this field contains Not Recorded.
Orig Owner	Name of the owner who created this resource. If created in a release prior to TDV 6.2, this field contains Not Recorded.
Last Modified Date	Date when the resource was last edited.
Last Modified User	Name of the user who last edited the resource.
Lock	Owner of the lock on the resource; or Not Locked.
Annotation	User-editable notes about the procedure.
Maximum Number In Queue	(All triggers except JMS triggers; in the Queue Properties section) Displays and lets you set the number of times a trigger can be queued simultaneously before duplicates are dropped. A value of 1 (default) means that the queue for this trigger can contain only the trigger currently being processed. If the trigger repeats before the previous one finishes its task, it is lost. When the value is set to 2 or greater, additional triggers start execution as soon as the current trigger finishes. Each refresh failure results in an email notification. A larger number of queues for a trigger reduces event loss but might use more memory. (JMS triggers; in the Queue Properties section) TDV sets this field to 0. You cannot change it, because no message queueing is done inside TDV.
Target Definition Set	(XQuery transforms) The path to the definition set containing the schema.
Target Schema	(XQuery transforms) The fully qualified name space for the schema.
Execute only once per transaction for each unique set of input values	(XQuery procedures) If other procedures are called multiple times with the same parameter and values from within a procedure, it is invoked only once and the original results are reused. This ensures that the execution results of a procedure are kept intact for the transaction, as long as the procedure's input parameter values remain the same. For details, see Setting Transaction Options for a Procedure, page 304.

To get information about a resource

1. Open the resource.
2. Select the Info or Configuration tab at the bottom of the resource editor.

Annotating a Resource

You can edit any resource that you own or for which you have editing privileges.

Whenever a resource has been edited or changed, Studio indicates this by enabling the Save button on the Studio toolbar, displaying the name on the tab editor in blue italic type with an asterisk, and displaying the editor tab in which the change was made in blue type with an asterisk.

Saving the resource removes the asterisk and displays the tab text in black type.

To annotate a resource

1. Verify that you have Read Access and Write privilege on the specific resource.
2. Open the resource.
3. Select the Info tab to view and modify annotations.

Moving, Copying, Deleting, or Renaming a Resource

You can move a resource by cutting a resource and pasting it into another location, or by renaming it, subject to the following rules:

- You cannot move an unpublished resource into Data Services, which only stores resources that have been published. See About Publishing Resources, page 408.
- You cannot move a published resource out of a Data Services container. You must delete it, and then, if you want, republish it.
- The only way to move the children of a data source is to move the entire data source.

- You can move a container (except a system-created container). When you do, all of the resources it contains are moved.
- You can move a leaf, such as a view or a procedure.
- As with any cut and paste sequence, if you cut a resource and do not paste it in its new location before you cut or copy another resource, the original resource is lost.

You can rename any resource in the resource tree. All references to the original filename in the SQL within resources in Studio are also changed to the new name. Resources with dependencies that are currently opened are closed to update and accommodate the changed dependency name.

To move a resource

1. Right-click the name of the resource and select Cut, or select the resource name and click the Cut toolbar button.
2. Right-click the name of the container in which to place the resource, and select Paste into; or select the container name and select the Paste into button on the toolbar.

To copy a resource

1. Right-click the name of the resource and select Copy; or select the resource name and click the Copy toolbar button.
2. Right-click the name of the destination container and select Paste into; or select the container name and select the Paste into button on the toolbar.

To delete a resource

1. Right-click the resource, and select Delete; or select the resource and click the Delete button.

   To select multiple resources for deleting, hold down Ctrl while selecting the resources one by one, or hold down Shift while selecting a set of adjacent resources.
2. In the Confirmation window, click Yes.

To rename a resource

1. Right-click the resource, and select Rename; or select the resource and then the Edit > Rename option.
2. Place the cursor anywhere on the resource name, type a new name, and press Enter.

Searching for Resources in the Server

You can search for any resource in the server using the Studio search feature.

The search field includes a timer that starts and waits about 0.3 seconds for you to stop typing. Each time you type a character, the timer restarts. When you stop typing, the server is queried for any resources whose name begins with the text you have typed, and resource references containing the resource name, path, and type are returned.

To search for a resource in the server

1. Select the Resource > Find Resource menu option.

   If you want to close the search window, click anywhere outside the window, or press the Escape key. Clicking the search window's title bar does not close the window.

2. In the text field, type the name of the resource you want to find.

   You can use a period to match a single character, and an asterisk or a percent sign to match any number of characters.

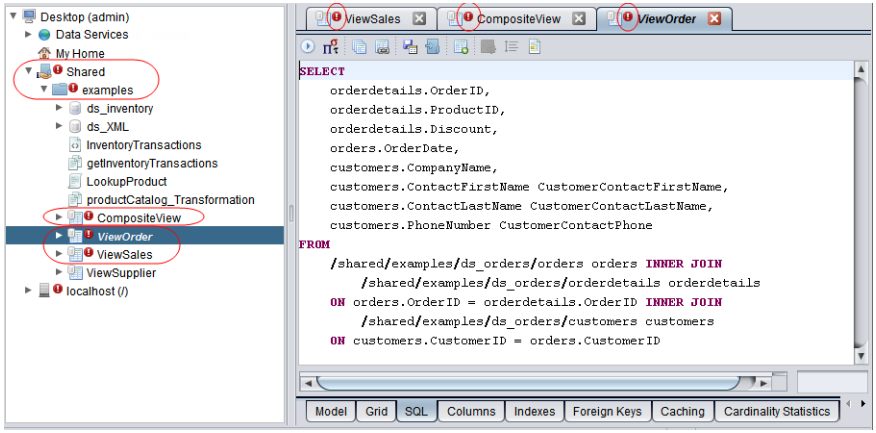
3. Use the up- and down-arrow keys to navigate through the list of resources.

4. Select the resource by clicking its name or by clicking the Enter key.

   The resource opens on the right, and on the left the resource tree expands to display the location of the resource.

Impacted Resources

An impacted resource in TDV is displayed with a red impacted icon in the Studio resource tree, because the resource definition is no longer valid. For example, if a composite view includes tables from a data source that is later deleted, the resource becomes impacted. The resource itself, any resource that depends upon it, and its containers display a red circle with an exclamation point in it.



This section contains:

Managing the Display of Impacted Resources

The name of an impacted resource always displays the impacted icon next to it. However, you can configure the parent folders of impacted resources to display or not display the impacted icon.

To manage the display of the impacted resource icon on folders

1. Open Studio.
2. Select Edit > Options.
3. Select the Warnings tab.
4. Select or clear Mark folders impacted when they contain descendants that are impacted.

5. Click OK.

For more information, see Customizing Studio, page 29.

Investigating Impacted Resources

There are multiple ways you can get the specifics about why a resource is impacted.

To discover the reason a resource is impacted

1. Discover the reason a resource is impacted in one of these ways:

- Hover your cursor over the impacted resource name in the Studio resource tree. Studio displays an error message in a popup.
- From the Resource menu, choose Impacted Resources. Studio opens an Impacted Resources dialog. If you expand the nodes, you can see all currently impacted resources marked with exclamation points in red circles.

  The Impacted Resources dialog helps you locate and investigate impacted resources:

  — Use the Expand All button to view all impacted containers and resources in the resource list.

  — Enter a search string in the Find field then click the Search button to view all resources with the search string in their name. If necessary, click the Expand All button to view the impacted containers and resources that match the search string.

  — Select or clear the radio button to display or clear the tree from the view of impacted resources.

  — Hover over any resource to get the explanation for why the resource is impacted as shown above.

  — Double-click any resource to open its editor.

  — Open the resource editor and click the tab for the type of resource:

  SQL view—Click the SQL tab.

  SQL script—Click the SQL Script tab.

  XQuery transformation—Click the XQuery tab.

  At the bottom of the panel, TDV displays an error message. You can see the entire error message by hovering your cursor over the error text box.

2. See Repairing Impacted Resources, page 49 to learn about the possible causes of the impact condition.

Repairing Impacted Resources

You can fix an impacted resource by correcting its definition. For example, if a dependent resource is missing, make it available. If there is a mismatch, make the resources match.

To repair an impacted resource

1. Locate the impacted resource and display its error message as described in Investigating Impacted Resources, page 48.
2. Review the table below to find the cause of the impaction.

Impact Condition	Cause
Configuration Error	The configuration of the resource is no longer valid, or its validity is uncertain. For example, when you import a data source, it might be impacted if it is unclear that the resources within the data source still exist. A reintrospection can resolve that uncertainty.
Missing Resource	A resource depends on another resource that no longer exists or is unavailable. For example, if a view references a table that was deleted, the view becomes impacted.
Mismatched Resources	A resource depends on another resource that has changed and no longer matches—for example, if the number of input variables changes for a procedure referenced in a view.
Mismatched Implementation vs. Interface	If the interface and implementation do not match, the resource is impacted—for example, if you design an interface using Design Mode in the view editor Columns panel and you add or delete columns.
Security Issue	A resource refers to another resource that you no longer have permission to access—for example, if the permission on a referenced data source is made more restrictive and you no longer have the required level of permission.
Syntax Error	A syntax error is introduced in the resource—for example, if you type an invalid SQL statement and save it.

Exporting (Archiving) and Importing Resources

You can save selected resources—or an entire TDV Server instance—to a TDV archive (CAR) file so that you can port the resources to another TDV instance or preserve a snapshot of TDV.

Archiving a resource is called *exporting*; deploying it to the same or another location is called *importing*. When you export a resource, a copy of the resource is archived into a CAR file and saved in a specified location, while the original repository configuration for the resource tree remains intact. Archived resources can be redeployed by importing this CAR file. This file is referred to as the export file or import file.

If you want to archive an entire TDV Server instance, it is best to first perform a full server backup. You can retain the resulting CAR file as backup, or use it to restore the TDV Server to a known state. See Using Studio for a Full Server Backup, page 60, for more information.

You can export and import resources using Studio or using TDV command-line utilities. See the *TDV Administration Guide* for information about using backup_export, backup_import, pkg_export, and pkg_import. With appropriate access rights, you can export a single resource, a group of resources, or the entire TDV configuration (except for local machine settings).

By default, importing adds CAR file resources to existing resources in identical container paths. Whenever both the resource path and name in the CAR file match the resource path and name on the target server, the CAR file version overwrites the target resource.

Access Rights for Export and Import

All objects can be exported by an administrator with all rights, or by an operations administrator with Backup (or Backup & Restore) rights who has Access Tools, Read All Config, Read All Resources, Read All Status, and Read All Users rights. Import requires all of the corresponding Modify rights.

- If encrypting resources when performing an export or import, you must communicate and share the encryption key that you used with others that are authorized to access those resources. There is currently no automatic way to securely share encryption keys between different TDV users.

When exporting:

- The user performing the export must have Read privileges on the selected objects and other associated rights.
- If the user performing the export has the Read All Users right, user profile definitions can be included (as in the pkg_export option -includeusers) in the export archive file. These items are disabled if the user does not have this right.
- A password for the CAR must be specified.

When importing:

- Resources captured in a CAR file can be imported to a TDV instance if the user has Write privileges on the target folders.
- Privileges are included among the items imported from a CAR file if the user performing the import has both the Modify All Resources and the Modify All Users rights.
- The password used to create the CAR file must be specified to import the CAR file. Or, if the CAR was created without a password, you can leave the password value blank.

See Rules for Importing Resources, page 54 for more information.

Exporting Resources

You can export selected nodes and resources in the Studio resource tree to a CAR file that contains the relevant resource definition metadata. Resources selected are exported with all of their child objects.

Exported CAR files can include caching configurations and the cached data, data source connections, access privileges, custom JAR files, dependencies, cardinality statistics, Discovery files, and the owners of the exported resources (called “required users”).

To export resources using Studio

1. Select one or more resources in the Studio resource tree.

   To select multiple resources for exporting, hold down Ctrl while selecting the resources one by one, or hold down Shift while selecting a set of adjacent resources.

2. Right-click and select Export, or choose File > Export <resource>.

   Studio displays the Export dialog.

3. For Name, type a name for the CAR file, or use the Browse button to locate a file in which to store the exported resources.

4. In the Include Resource Information section of the Export <resource> dialog box, select and uncheck export options as required.

   Each export option is enabled only if the user has the appropriate rights for it.

Export Option	Description
Caching	Resource caching configurations: storage type and location, scheduling information, materialized view data.
Privileges	Resource group and user privilege information. User privilege information for exported resources is imported into the target TDV installation only for matching user, group, and domains on the target, and only if the user performing the import has the Modify All Users right or the Modify All Resources right.
Dependencies	Any dependency resources in the exported CAR file. To view dependencies that have been exported, open the resource and select the dependency panel.
Required Users	Owners of the exported resources.
Data Source Connections	Resource-specific connection profiles and configurations, including host, port, sign-in, and password. This option requires the Read All Resources right.
Custom Jars	JAR files created to support new data sources. JAR drivers in the conf\adapters\system folder are exported, but JAR files in the apps\dlm\<DATASOURCE>\lib folder are not exported, regardless of this setting. Custom JAR settings overwrite system JAR settings, which in turn overwrite DLM JAR settings.
Cardinality Statistics	Includes data source statistics, and configurations such as refresh mode, refresh timeframe, manual cardinality overwrites, and specific column settings.
Discovery Files	Be sure to include Discovery files, such as indexes and relationship files.

Choosing check-box options in this section is equivalent to enabling options using the command line in the pkg_export utility. For more information on the pkg_export utility, see the *TDV Administration Guide*.

5. Type a password to encrypt the CAR file. Type the same password to confirm it.

   A password is required to export resource data.

6. Optionally enter a description of the export file in the Description box.
7. Click OK.

Export and Import Locked Resources

When you export resources using Studio, the selected resource definitions and containers are exported regardless of whether they are locked or not. However, an export using Studio excludes the locking information in the exported CAR file metadata. If you want to preserve the lock status and ownership, you must export your resources using one of these methods:

- Perform a full server backup. See Using Studio for a Full Server Backup, page 60.
- Use the **`backup_export`** command-line utility. See the *TDV Administration Guide* for more information.

The export CAR file contains all pertinent lock metadata. If you use the backup_import utility to import resources, and the CAR file contains lock metadata (evidenced by the presence of a file named resourcelocks.xml), the resources are restored with the locks intact. Importing from Studio menus brings in the resources but not the lock metadata.

Marking Rebindables

Many resources, such as views, procedures, and Web services, are dependent on one or more underlying resources. Dependent resources are considered 'bound' to the underlying resources. Imported resources often need to be bound again in their new location.

Binding a resource to its dependencies is also often necessary in the import process. For example, a procedure P1 might retrieve data from table T1 in the development data source DS1, and refer to it accordingly in its FROM clause. In this case, P1 is said to depend on T1, and P1 and T1 are said to be bound. When importing P1, the user must ensure that the connection to T1 is fed from either DS1 or another data source—for example, a live production data source rather than a QA or development data source.

For additional information, see Rebinding a View, page 248 and Rebinding a Procedure, page 303.

Exported resources are often bound to underlying sources, and the binding should be reestablished (or changed) when the resources are imported.

When a resource is marked as rebindable in the CAR file, Studio import of this resource displays the path and description in a dialog box that lets the user specify a new path for the dependency resource.

To mark rebindables with a command-line program

1. Use the command-line pkg_export program.
2. Specify the option to mark the underlying source as rebindable:

```
-rebindable <SourcePath/.../ResourceName> <Description>
```

Or

1. Use the command-line pkg_import program.
2. Specify the following command:

```
-rebind <OldPath> <NewPath>
```

Rules for Importing Resources

Importing follows certain rules to resolve conflicts during import. These rules are:

- If an imported resource does not exist, it is created. The person performing the import is granted privileges as the creator (such as Read | Write for a folder or Read | Write | Execute for a procedure). If the user is in the admin group and has requested include access, the resource reverts to its prior owner, and any privileges in the import are also put in place.
- If a resource is imported to a nonexistent folder, the folder and any parent folder that does not yet exist are created, and the user performing the import is assigned Read | Write privileges on and ownership of these folders.

  Note: Missing folders are not auto-created in the Data Services folder.
- Importing a pre-existing resource overwrites the older version (assuming the required Write privileges), except that:
  - The owner is not changed.
  - User privileges are not overwritten unless explicitly changed by the import. For example, if Jean has Read | Write permissions and Sue has Read | Write permissions, and the import file lists Jean as having Read

permission but not Sue, Jean's privileges are updated to just Read, but Sue's are left intact.

— If the resource is a folder or data source, its child resources are not removed.

Restrictions when importing resources

- Configuration settings are not carried over when you export or import a resource using Studio.
- Resource creation in a folder requires Write privilege on the folder path leading to that folder.
- A resource cannot be overwritten without Write privilege on that resource.
- You cannot export just part of a physical data source; you must export all of it or none of it.
- You cannot import just part of a physical data source; you must include the source definition itself.
- The Data Services folder enforces strict structure rules.
- You cannot import anything that was exported from the Data Services folder to a location outside of that folder.
- You cannot import anything that was exported from outside the Data Services folder into that folder.

Importing a Resource

You import CAR files to replicate TDV resources and resource configurations. CAR files are created by exporting TDV resources, and they usually contain many object resources—possibly even a full server backup. For details on creating a CAR file, see Exporting Resources, page 51 and Using Studio for a Full Server Backup, page 60.

During the import process, you can choose to import or not import caching information, privileges, custom adapter JARS, and data source connections. You can also override locks, overwrite resources, merge folders, and create caching tables.

Resources imported from an archive using Studio are placed in the selected folder in Studio's resource tree. Import performs a path relocation on all folder path roots in the archive. See Rules for Importing Resources, page 54 for details about the rights and privileges related to importing resources.

Prior to performing the import, you can view information about the archive file to make sure it is the one you want. You can also preview the messages generated during the import without actually performing the import.

To import a resource

1. Right-click the container into which you want to import a resource, and select Import; or select the resource and then the option File > Import Into <container>.

   The Import into window opens.

2. Use the Browse button to locate and upload the CAR file, or type the full path to the import file in the File field.
3. Optionally, type a password. If the CAR was created using a password, you must type that password here. If the CAR was created without using a password, you can leave this field blank.
4. Clear items in the Include Resource Information section that you do not want to import.

   Note: A resource information box is checked only if that resource is present in the CAR file *and* you have appropriate user rights. For example, the Privileges check box is enabled only for users with the Read All Users right.

Option	Description
Caching	Includes or excludes resource caching configurations, including storage type, location, scheduling information, and materialized view data.
Privileges	Includes or excludes resource group and user privilege information. User privilege information is imported into the target TDV installation only if matching username, group, and domains exist on the target and the user performing the import has the Modify All Users right or the Modify All Resources right.
Override Locks	Clears existing resources from the target and sets the TDV Server to the resource definitions present in the full server backup CAR file. The importing user must have the Unlock Resources right to use Override Locks.
Create Caching Tables	If selected, the cache status, tracking and target tables required by cached resources are created, if not already present in the database for your data source.

Option	Description
Overwrite	Overwrites or stops import of resources that have identical names and folder paths. Owner and user privileges are not overwritten unless they are explicitly changed to new privilege setting. Use this option to guarantee that the new TDV workspace matches the folders and content present in the CAR file. This option clears target folders before copying the CAR file contents to them.
Custom Jars	Includes or excludes any custom JAR created to support a new data source.
Merge Users Folder	Imports all users present in the CAR file who are not already present in the server target. This option takes precedence over the Include Users Folder option. This option's behavior might be altered by the Overwrite option.
Data Source Connections	Includes or excludes resource-specific connection profiles and configurations, including host, port, sign-in, and password. This option requires the Read All Resources right.
Include Users Folder	Includes or excludes resource owners who might be present in the CAR file. Imports all users present in the exported CAR file unless Merge Users Folder is specified. By default, domain, group, and user information are not included in export or import packages. When users are not included, the importing user becomes the owner of the resource.
Include Cache Policies	Includes or excludes cache policies that might be present in the CAR file. If your CAR file includes cache policies, import the CAR from the Studio Desktop directory level. Cache policies must be save and used from the /policy/cache Studio directory.
Discovery Files	Check it to include Discovery files (such as indexes and relationship files) in the import.

Different combinations of Overwrite, Include Users Folder, and Merge Users import options can alter the user definitions, resource ownership, and usage privileges defined on the TDV target after the CAR-file import is complete.

This table shows all possible combinations of these options. Check marks on the left side of the table show the import options used, and check marks on the right side show the results on the TDV target after the import.

Import options used			Users imported into CIS target			
			From *.car file		CIS target	
Overwrite	Include Users	Merge Users	UserA	UserB (car)	UserB (CIS)	UserC
√					√	√
√	√		√	√		
√	√	√	√	√		√
√		√	√	√		√
	√		√	√		√
	√	√	√		√	√
		√	√		√	√
					√	√

Where a checkmark indicates an option used or the users defined on the CIS target after the pkg-import with the given options is performed.

Note: If Overwrite and Include Users Folder are both checked, all existing user definitions are removed and replaced with the users present in the import CAR file. The user performing the import must have Read and Modify All Users rights to use this combination.

Choosing check box options in this section is equivalent to enabling options in the pkg_import command-line utility. See the *TDV Administration Guide* for more information.

5. If you edit the entry in the File field, click Refresh to view the following information about the import file:
   - File Type—archive type (whether partial or the entire server)
   - Date—day, date, and time when the resource was exported
   - User—name of the user who exported the resource
   - Server—host machine and port from which the resource was exported
6. Select the Show Rebinding Options box to bind a resource to an underlying source for the first time. (For details about resource binding, see Marking Rebindables, page 53.)

   If you check the Show Rebinding Options box the Import Into rebinding window appears. See Rebinding an Imported Resource, page 59.
7. Optionally, click Preview to view a list of the contents of the import file without actually performing any imports.

   The Preview button works like the -messagesonly option to the pkg_import command-line utility. This option displays the messages generated in a package import without actually performing the import.

8. Optionally, type search criteria in Find, and click Search. Use the up or down arrow key to jump to the next instance of the search text. The text that you are searching for is highlighted in yellow.
9. Click Import to import the resources.
10. Click Done to complete the importing process.

Rebinding an Imported Resource

A view depends on one or more underlying sources, and the view is considered bound to those underlying sources. A view can be bound to tabular data or a procedure. Rebinding is useful when:

- You create a view with its sources and later decide to rebind the view to different sources.
- An execution error has occurred for a view because a source with which the view was initially bound does not exist any more.

If you checked Always Show Rebinding Options in the Import window, the window shown below appears, listing the resources that need to be rebound.

To rebind an imported resource

1. Follow the procedure to import a resource described in Importing a Resource, page 55 through step 5.
2. Select the resource to rebind in the upper part of the window.
3. Use the Browse button or type the Current Path to the resource.
4. Use the Browse button or type the New Path to which to bind the resource.
5. Click Rebind to rebind the resource.

   Note: If Hide rebound item is checked (default), each item disappears from the top part of the window as it is rebound.
6. When all resources have been rebound, click Next.

   The Import Into window appears, the same as when no resources need to be rebound.
7. Click Done to close the Import Into window.

Using Studio for a Full Server Backup

Studio can perform a full TDV Server backup. This capability is available for those with these administrative rights: Access Tools, Read All Resources, Read All Users, and Read All Config.

This option is equivalent to the backup_export command-line utility.

To perform a full server backup

1. Select the Administration > Full Server Backup menu option.

   The Full Server Backup window opens. Check boxes in the Include Resource Information area are:

   — Checked and dimmed if the information must be included for a full server backup.

   — Unchecked and dimmed if the information cannot be included in the backup.

   — Checked if the information is included by default but can be excluded.

   — Unchecked if the information is omitted by default but can be included.

2. Check or uncheck the undimmed boxes as appropriate to this backup.
3. Use the **Browse** button to specify a file in which to store the exported resource.

   The **Save** window appears for specifying and locating an export file. You can:

   — Create a new export file by typing a name for it in the **File name** field. The filename extension (.CAR) is automatically added.

   — Use an existing CAR file. Locate it using Browse, and click **Save**.

   If you use an existing CAR file, its contents are overwritten when you complete the export process.

4. Type a password to encrypt the CAR file. Type the same password to confirm it.

   A password is required to export resource data.

5. Optionally, type a description for the export file in the Description field.
6. Click **OK**.

Working with Resource Locks

In Studio, a resource can be opened, locked, and modified starting with a right-click on the resource name in the Studio resource tree.

You can also lock and unlock all resources in TDV using a single configuration parameter. When Studio resource locking is enabled with this configuration parameter, resources must be unlocked before they can be edited. Studio resource locking is disabled by default. See the *TDV Administration Guide* for more information.

When a resource is locked, other developers working on the same TDV Server instance (or active cluster) can see that a lock has been placed on a resource or a resource tree node.

Note: Resource locking is not intended as a security feature. An administrator can protect defined resources by managing rights and privileges.

Administrators with the Modify All Users right, and resource owners, can revoke the Write privilege (modify) for specific resources, thereby preventing anyone from modifying those resource definitions.

Constraints on new resources in locked containers include the following:

- Only the user who locked a resource can modify that resource or save changes to it.
- Only the lock owner can create or destroy resources within a locked container.
- All resources within a locked container inherit the lock from the parent container, which in turn belongs to the original lock owner.

This section includes:

Locking and Unlocking a Resource

You can lock a resource in Studio to prevent other users from changing it. The resource owner or an administrator with the Unlock Resources right must unlock a resource before other users are allowed to save changes that resource. Unlocking a tree of resources also unlocks all objects in that tree.

If the user has the Unlock Resources right, all other locked objects for which the user has the Read privilege are shown and selected for unlocking.

To lock a resource

1. Verify that you are the resource owner.
2. Select one or more resources or containers in the resource tree container, right-click, and select Lock Resource.

To unlock a resource

1. Verify that you are the resource owner or that you have the Unlock Resources right.
2. Save the resource prior to unlocking it.
3. Right-click the locked resource and select Unlock Resource.

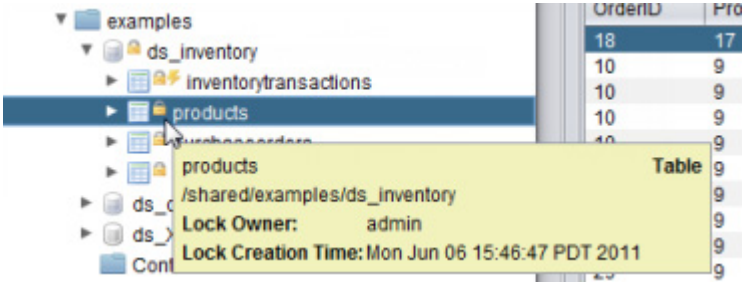
   All resources that can be unlocked by the current user are shown.

Identifying Locked Resources

You can identify locked resources in the resource tree by noticing the small lock icon next to the resource icon. All Studio instances connected to the same TDV Server or Active Cluster have the same set of locked icons for resources for which they have Read privilege access. In the Lock field on the Info tab, the type of lock (implicit or explicit), who created the lock, and when it was created is displayed. The lock type is explicit if the user set a lock on the specific resource or implicit if the resource inherited the lock from a parent container.

To get detailed information on a locked resource

1. Hover the cursor over the resource name to see the lock status, for example.



   — Lock Owner—User who set the lock

   — Lock Creation Time—Time when the lock was set

2. To learn about the lock type, open the resource and select the Info tab.

Change History of a Lock

The server events log, cs_server_events.log, captures change history for resource locks. It contains the username and domain of the person who locked or unlocked the resource, the time stamp, the data resource, and any annotation entered at the time of unlocking.

You can monitor lock change events using a third-party SNMP utility. SNMP IDs 20805 (csResourceLock) and 20806 (csResourceUnlock) reveal locked resources and changed resources, with change descriptions entered when the highest-level resource node was unlocked.

For more information on SNMP monitoring, see the *TDV Administration Guide*.

Working with Data Sources

TDV lets you work with a wide variety of industry data sources, including relational databases, file data sources, WSDL data sources, application data sources, and so on, by providing built-in data adapters that help you define and configure the data sources for TDV. This topic describes how to work with data sources in TDV.

About Data Sources and TDV

TDV integrates your data sources into a single virtual data layer that can be used to model rich information resources. You can add many types of data sources to the TDV platform using Studio, defining connection profiles, specific source capabilities, and detailed source introspection metadata. TDV views and procedures built on the data modeling layer can take advantage of capabilities unique to each data source, allowing you to integrate those disparate parts into a single virtual data layer.

Adding a data source means creating a metadata representation of the underlying native data source in the TDV repository. It does not mean replication of the data or replication of the source. The data source metadata for each data source type includes things like how the data source:

- Defines and stores data, including catalogs, schemas, tables, and columns.
- Accepts and responds to requests for data.
- Handles insert, update, and delete transactions.
- Executes stored procedures.
- Makes data-related comparisons.

A virtual layer of information about data source capabilities allows the TDV Server Query Engine to create efficient query execution plans that leverage data source strengths and inherent advantages of preprocessing data at the source.

The following sections elaborate on the preparation and use of data sources:

About TDV Adapters

TDV Adapters simplify and accelerate high-performance access to a wide range of data sources including popular enterprise applications, relational and multidimensional data sources. TDV adapter features include interactive metadata mapping, integration of and resolution for most vendor-specific SQL, and statistical analysis of source tables to enable quick optimization of query execution plans. TDV adapters provide certified, vendor-specific connectivity that leverages data source APIs. TDV adapters for non-relational data sources let you create views that represent data as relational tables.

TDV adapters capture data source settings that are then recorded in the metadata modeling layer, for use when query execution plans are calculated. Refer to Adding and Removing Data Source Resources, page 74, for more information.

TDV provides some data adapters that have default settings. If data source instances have been changed to handle data and SQL in a special way, you might need to modify the adapter for best performance.

About Introspection

Introspection is the process of retrieving metadata and data from your data sources. For more information, see About Introspection and Reintrospection, page 190.

About Federated Queries

Data virtualization makes it possible to execute federated queries, which are queries against multiple data sources. Each of these data sources may be of a different type, or of the same type with different settings. In addition, TDV can execute any of almost 300 functions natively when required, making TDV itself a source of federated query results.

The following table lists where in the documentation you can find further information about such issues.

Federated-Query Topic	Where Discussed
Data source function support (push vs. no-push)	Listed by data source: "Function Support for Data Sources" section of the *TDV Reference Guide.* Listed by function: "Function Support Summary" section of the *TDV Reference Guide.*
Suppressing push of SQL operators	The five DISABLE_PUSH sections of *TDV Query Engine Optimizations, page 593*.
Pushing custom functions	Pushing a Custom Function to the Native Data Source, page 274.
TDV function support	"TDV Support for SQL Functions" section of the *TDV Reference Guide.*
Case-sensitivity and trailing spaces in comparisons	See the *TDV Administration Guide*. "Case Sensitivity and Trailing Spaces" section, CASE_SENSITIVE and IGNORE_TRAILING_SPACES option sections, and Case sensitivity and Ignore Trailing Spaces configuration parameters in the *TDV Reference Guide.*
Miscellaneous function support issues, such as data precision, truncation vs. rounding, collation, time zones, and interval calculations	"Function Support Issues when Combining Data Sources" section of the *TDV Reference Guide.*

Data Source Categories

TDV supports the following general types of data sources:

Type	Description
Relational Data Sources	See the *TDV Installation and Upgrade Guide* for supported data sources. Specific connection details, parameters, and settings are slightly different for each adapter. For more information on using these, see Configuring Relational Data Sources, page 77.

Type	Description
File Data Sources	See the *TDV Installation and Upgrade Guide* for supported data sources. For more information on using these, see Configuring File Data Sources, page 165.
WSDL and Web Services Data Sources	TDV supports Web Services Definition Language (WSDL), SOAP, and REST. Web services can be "bound" to any message format and protocol, but these bindings are the most popular: SOAP, REST, HTTP, and MIME. TDV supports several profiles for SOAP binding, including the following: SOAP over HTTP, SOAP over WSIF JMS, and SOAP over TIBCO JMS. See Configuring Web-Based Data Sources, page 105 for more information.
LDAP Data Sources	You can use LDAP sources as data sources or (in some cases) as authentication servers. The TDV LDAP basic adapter supports the LDAP v3 protocol, and it is used when you want to use an LDAP source as an introspected data source. Most LDAP sources that are LDAP v3 compliant can be used with the TDV LDAP driver to connect and introspect the selected resource nodes for use in the modeling layer. When you want to use an LDAP source as an authentication service, the TDV Web Manager > Domain Management page is used to add and initially configure the LDAP domain. See the *TDV Installation and Upgrade Guide* for a list of compatible LDAP directory services. Refer to the *TDV Administration Guide* section on LDAP Domain Administration for more details on how to implement an LDAP domain for TDV user and group authentication. See Adding an LDAP Data Source, page 174.

Adding a Data Source

This section is an overview of the process for adding a data source. It includes the following topics:

- Custom Adapters, page 70).
- Auto-Discovering Data Sources, page 71

To add a data source

1. In the Studio navigation tree, select the directory in which you want the new data source to reside.
2. Right-click and choose New Data Source from the popup menu, or choose the File > New > Data Source option.

   Studio opens the New Physical Data Source dialog. This dialog displays currently supported and custom-defined data source adapters in the Select Data Source Adapter pane.
3. Optionally, if one of the built-in adapters do not fit your data source, you can create a new, custom adapter (see Custom Adapters, page 70).
4. Optionally, if you are unsure of the data sources you want to define or that are available to you on your network, you can auto-discover resources (see Auto-Discovering Data Sources, page 71).
5. Search for and select the adapter to use to connect to your data source, and click Next.

The configuration window that is displayed depends on which data source adapter you have selected.

6. Provide the data source connection information to configure your data source:
   a. For Datasource Name, enter a unique user-defined name for the introspected data source. It can be any name that is valid in TDV. This name will be displayed in the resource tree when the data source is added to the TDV Server.
   b. For Connection Information on the **Basic tab**, supply the required information. Required information will vary depending on the type of data source you are adding.

   — Username and Password—Valid username and password to your data source.

   — Save Password and Pass-through sign-in —These fields work together. Discussion for how they work is in the *TDV User Guide*.

   c. Click the **Advanced tab**, specify details about the connection.

   Some properties refer directly to the configuration of the data source Server and must be provided by an Administrator. Other properties are specific to TDV and how it interacts with the data source.

   — Connection Pool Minimum Size, Maximum Size, Idle Timeout(s), and Maximum Connection Lifetime—These control the JDBC connections

made to the data source, specifying timeout in seconds, the minimum and maximum number of simultaneous connections in a connection pool.

— Connection Validation Query—A simple query to test whether the connection is valid or not. You can leave it empty, or use simple queries like "select * from dual."

— Organization Id Cache Timeout Seconds—Length of time to cache user security information. After this time the security information is refreshed from the data source. If a user's security information is changed before the cache is refreshed, the changes will not be available to the Suite data source until the cache is refreshed.

For the specific connection information needed to configure each data source see:

— Configuring Relational Data Sources, page 77

— Configuring Web-Based Data Sources, page 105

— Configuring File Data Sources, page 165

For information on configuring any of the Advanced Data Sources, see the additional adapter guides that are provided as online help or in PDF form.

7. Create the data source using one of these buttons:

— Click Create & Close to create this data source which can be introspected later.

— Click Create & Introspect to select the data resources and introspect them. See Retrieving Data Source Metadata, page 189 for how to do this.

Custom Adapters

You can create a new custom adapter with the New Adapter button on the right side of New Physical Data Source dialog. Both system adapters and custom adapters that were created earlier are listed in the Select Data Source Adapter list in the left panel. There are specific reasons why you might want to create a custom adapter. See the *TDV Extensibility Guide* for more information see Adding and Removing Data Source Resources, page 74.

See the Extensibility Guide to see what capability changes might require use of a custom adapter.

Auto-Discovering Data Sources

TDV provides an auto-discovery feature that automatically discovers and displays relational data sources that are visible on the local area network. If a large number of relational data sources and servers are on the network, the scan results might take a long time to complete. It is usually more efficient to add data sources manually. See Adding Relational Data Sources, page 85.

To auto-discover data sources on your network

1. Choose the File > New > Data Source.
2. In the New Physical Data Source dialog, click Auto Discovery.

   The dialog for scanning the network for data sources opens.
3. Accept the default name displayed in the Machine Name field or type the name of the computer from which to scan the network, and click Scan Network.

   TDV searches for data sources (using data source default port settings) on the network and displays a list of host computers where specific data sources are visible to the machine specified in Machine Name.
4. Select a machine name (or IP address) and click Next.

   Each scan allows for a single addition of a relational data source.
5. For further details, see Adding Relational Data Sources, page 85.

Editing or Reviewing Configuration Information for a Data Source

After a data source has been configured and introspected as a new physical data source, the configuration information can be reviewed by opening the data source in Studio. The tabs display details about the data source connection and other configuration information. The details differ depending on data source type (relational, web-based, file, and so on).

If you make changes, you need to save changes when you close the data source. You also need to reintrospect the data source to apply your changes. For more information, see Reintrospecting Data Sources, page 206.

To edit or review data source configuration details

1. Right-click the data source name in the Studio resource tree and select Open.
2. On the Configuration tab, depending on your data source type, you can review and modify the following information:
   — General Options

General Options	Description
Enable Data Source	Check box to enable or disable the data source for use.
Add/Remove Resources	Provides access to the re-introspection dialog.
Test Connection	Provides a way to validate the connection definition for the data source.

   — Caching

Caching Options	Description
Status Table	Tracks which views and procedures currently have cached data stored, when they were last refreshed, and so on.
Tracking Table	Tracks which views and procedures are currently using the data source for caching, and what tables in the data source are in use.

   — Connection information Option

Connection information	Description
Basic tab	Displays the basic connection details TDV uses to connect and sign in to the data source.
Advanced tab	Displays the advanced connection details TDV uses to connect and sign in to the data source.

3. Review or edit the following information on the Info tab:

Field	Description
Name	User-defined name given to the data source during introspection.
Type	Type of the resource, which is Data Source in this case.

Field	Description
Owner	Domain to which the owner of the data source belongs, or the sign-in name of the user who created the data source.
Orig Creation Date	Date the data source was originally created.
Orig Owner	Name of the owner who originally created this data source.
Last Modified Date	Date when the resource was last edited.
Last Modified User	Name of the user who last edited the resource.
Lock	Owner of the lock on the resource, or Not Locked. TDV resources can be locked by a user to prevent concurrent and conflicting changes. See the Annotating a Resource, page 44.
Cluster Health Monitor Table	Displays the table used to monitor the health of an Active Cluster. It contains the heartbeat messages from all nodes. Its purpose is to help caching determine the health of other nodes in the cluster in case of a cluster split. Refer to the *Active Cluster Guide* for more information.
Refresh Capability Information	Use to refresh the capability information for this data source.
Case Sensitivity	Reports whether the case sensitivity and trailing spaces settings do or do not match between TDV and the data source capability file. A mismatch does not necessarily reduce performance unless it results in a full-table scan performed locally by TDV. See the "Configuring Case and Space Settings" topic in the *TDV Administration Guide*.
Trailing Spaces	
Annotation	Area at the bottom of the panel that displays notes about the data source. Any user with Write permission can edit the annotation.

4. Save any changes.
5. If changes were made, perform a re-introspection.

Viewing Data Source Table Details

Data source tables introspected in TDV can be opened to view the table's metadata attributes.

To view an introspected data source table

1. In the Studio resource tree, navigate to a data source table and open it.

   Studio displays the Columns panel for the table that is open in the workspace.

2. The first three columns in the table are provided for every introspected data source table.

Column	Description
Name	Column name. A key icon to the left of the name indicates that this column is a primary key in this table.
Type/Reference	Displays the data type applied to this column by TDV.
Native Type	Displays the native data type of this column in the original data source.

3. See the *Discovery User Guide* for information about the additional columns.

Adding and Removing Data Source Resources

You can add or remove data resources at any time. Adding resources requires introspection of the data source to obtain the metadata about the new resources.

When you remove resources from TDV, you are deleting the TDV metadata about the resource and not the actual resource. If you remove a resource that had dependencies, the dependent resource names in the Studio resource tree turn red, alerting you to resource problems.

To add or remove data source resources

1. Right-click the data source in the resource tree, and select Add/Remove Resources, or open the data source's Configuration tab and click Add/Remove Resources. See Editing or Reviewing Configuration Information for a Data Source, page 71.

   The Enter Password window opens if the password to access the data source was not saved when the data source was originally added to the server. For further details, see the descriptions for the Save Password and Pass-through sign-in fields under Adding Relational Data Sources, page 85.

2. If requested, supply the password that is required to access the data source, and click OK.

3. In the Add/Remove Resources window, do one of the following:
   - To add a resource, select the corresponding check box.
   - To remove a resource, clear the corresponding check box.
   - To change the settings for adding new resources during reintrospection, check the Detect New Resources During Re-Introspection box. For details, see the section Setting Introspection Filters on a Resource, page 203.
4. Click **Next** to review the Introspection Report and if satisfied begin the introspection.
   - This window can be dismissed at any time to continue running in the background. The re-introspection process also runs independently of the Studio session.
   - Those resources that do not have a change in the metadata used by TDV contribute to the Skipped count.
   - Use the Show pull-down filter to quickly show any errors or warnings.
   - The introspection (re-introspection) status report can be exported as a text file.

Testing the Connection to Your Data Source

After adding a data source, you can test whether the connection details you supplied are working.

To test the connection

1. Right-click the data source in the resource tree, and select Open.
2. In the editor that opens on the right, select the Configuration tab, and click Test Connection.

   The Connection OK message appears if you have a successful connection.

If the connection fails, examine the error message and review the following common remedies.

Common Problem	Suggested Remedy
Connection failure.	Verify that the TDV host has access to the Internet through HTTP (port 443, 80). If a Web browser can be used to reach the data source but TDV cannot, check the Web browser's configuration for Web proxy settings and enter them into the Advanced Properties of the data source.
Connection OK.	The connection was successful.
Password expired.	Login to the data source through its Web site and reset your password. If your organization's password policy forces users to change their passwords every certain number of days, you should enable the Password Never Expires option on your profile.
The SAP account is not authorized for API access.	Contact SAP or your organization's SAP administrator for assistance in enabling your organization's account for API access.
The Siebel account is not authorized for API access.	Contact Siebel or your organization's Siebel administrator to have your organization's account enabled for API access.
Username and password are invalid or locked out.	Verify that you have provided the correct user name and password by logging into the data source's Web site itself. Contact your organization's Administrator for assistance.

Configuring Relational Data Sources

This topic describes the configuration options that apply generally to relational data sources for connection with TDV. Follow the steps in Adding Relational Data Sources, page 85 for every relational data source.

Data type and function support are described in the *TDV Reference Guide*.

For details on versions supported for each type of data source, see the *TDV Installation and Upgrade Guide*.

The following topics are covered:

About Pass-Through Login

Simple implementations of pass-through do not support multiple pass-through passwords for a single user. Each user session may use only one user sign-in and password to create new connections that have data sources enabled with pass-through login. If pass-through login is enabled on multiple data sources, connection authorization must be based on a single user login-password pair for each user, or the connection to the data source fails for federated queries.

TDV Server and JDBC clients can support pass-through login for multiple data sources, but the data source credentials passed from the end-user JDBC client must be set explicitly for those data sources. See "Setting Pass-Through Credentials for JDBC Clients" in the *TDV Client Interfaces Guide*. The JDBC setDataSourceCredentials method is used to register the data source pass-through credentials for each data source to be so used.

Each client making a request for data from pass-through-enabled data sources has to establish a connection, which requires significant processing time.

In pass-through mode:

- If you save the password, you can introspect without resupplying the password.
- You can perform the following operations even if you did not save the password:
  - Query, update, insert and delete operations (but you need to resupply the original login credentials for the current session).
  - Reintrospect, add, or remove data source resources. You are prompted to resupply the password that was used when the data source was originally introspected.
- You cannot perform the following operations if you do not save the password:
  - Schedule reintrospection
  - Gather statistics using the query optimizer
  - Perform scheduled automated cache updates

Note: Pass-through authentication is not allowed for TDV-built system-level accounts.

When using scheduled cache refreshes, you must specify a login-password pair in the data source, and the pair must be the same for the resource owner (or creator) who is accessing TDV and the target data source. If this is not the case, the cache refresh fails.

If a view uses a data source that was added to TDV Server using pass-through mode, and the password has not been saved, row-based security may affect the cache refresh. For example, a cached view named CachedCommonView uses the SQL statement SELECT * FROM db2.T1; user John is allowed to view only 10 rows; user Jane is allowed to scan 20 rows. Whenever Jane refreshes the view cache, both Jane and John are able to view 20 rows, but whenever John refreshes the view, Jane can view only 10 rows.

The operations you can and cannot perform in pass-through mode depend on if Save Password is checked as follows:

Save password?	Operations you can perform	Operations you cannot perform
Yes	Introspection. You do not have to re-supply the password.	N/A

Save password?	Operations you can perform	Operations you cannot perform
No	• Query/update/insert/delete operations. You need to resupply the original login credentials for the current session. • Reintrospection, Add/Remove data source resources. You will be prompted to resupply the password that was used when the data source was originally introspected.	• Schedule reintrospection. • Statistics gathering, using the query optimizer.

About Data Source Native Load Performance Options

Most data sources have proprietary ways to optimize performance, and they are typically enabled by default. If TDV detects that your operation (for example, a direct SELECT or INSERT) can be made faster using a native loading option, TDV attempts to use it.

Depending on the data source type and the TDV feature you are using, native load performance options vary as follows.

Data Source Type	INSERT and SELECT	Caching and Data Ship Performance Options
DB2	Native load with INSERT and SELECT is supported.	• Native load with INSERT and SELECT • Native and bulk load with LOAD utility
Microsoft SQL Server		• Bulk loading using BCP
Netezza	Native load with INSERT and SELECT is supported.	• Native load with INSERT and SELECT • External tables
Oracle		• (Version 9 and 10) Native load with INSERT and SELECT. (Native load with DB link is not supported.) • (Version 11) Database links
PostgreSQL (Greenplum)		• COPY

Data Source Type	INSERT and SELECT	Caching and Data Ship Performance Options
Sybase IQ		• Location • iAnywhere JDBC driver
Teradata	Native load with INSERT and SELECT INTO is supported.	• Native load with INSERT and SELECT • FastLoad/FastExport
Vertica		• Bulk load utility • Export to another Vertica database • Native load with INSERT and SELECT

Data Source Limitations

Apache Drill Data Source Limitations

if the CAST function uses any non-table column as an argument, then the database returns an empty result set through JDBC. Non-table columns are for example, NULL or string literal.

For example, it is best to rewrite the following query:

```
select cast(null as varchar) a13
from /shared/postgreDrill/DRILL/"postgres.ga"/all_datatype_1k as
table1
```

As:

```
select cast(null as varchar) a13, id
from /shared/postgreDrill/DRILL/"postgres.ga"/all_datatype_1k as
table1
```

DB2 and DB2 z/OS Data Source Limitations

See the IBM DB2 Universal Database client documentation for more information on installing and connecting with DB2 instances.

Hive Compatible Data Source Limitations

Apache Hive is a data warehouse infrastructure that lets you query and analyze large data sets stored in Apache Hadoop files. Hive uses a simple SQL-like query language called QL to query the data. TDV supports several Hive-based database solutions. For a list and associated version numbers, see the *TDV Installation and Upgrade Guide*.

Because queries are distributed across several Hive nodes, queries that process only a small amount of data might finish slower than expected.

Information about how to enable security on your Hive data sources is described in the *TDV Administration Guide*.

Informix Data Source Limitations

Informix stored procedure parameters cannot be reliably introspected. To work around this, manually define the parameters using the Design Mode in the Parameters panel. See Editing a Stored Procedure in an Introspected Data Source, page 299.

Microsoft SQL Server 2008 Limitation

There is a known problem with Microsoft related to JRE 1.8.0_172 that results in disabling of the 3DES_EDE_CBC transport layer security algorithm. If you encounter this problem, you can enable 3DES_EDE_CBC in <TDV_install_dir>/jre/lib/security/java.security.

To re-enable 3DES_EDE_CBC

1. Navigate to the <TDV_install_dir>/jre/lib/security/java.security.
2. Open the file and remove 3DES_EDE_CBC from the jdk.tls.disabledAlgorithms setting.
3. Restart the TDV Server.

Netezza Data Source Limitations

- Placeholder parameters (designated by question marks) in prepared statements cannot be evaluated for use with the Netezza data ship optimization, because the variable must be resolved prior to submission.
- Federated queries with database-specific functions must be able to push that SQL directly to the data source, or the query will fail.
- Netezza UDA and UDF need to be assigned full resource names so they can be used in queries.
- Composite views can invoke Netezza UDA or UDF using full TDV resource names of the following general form: /users/composite/admin/Netezza/Aggregate/Builtin/"COUNT"()
- Each query (including the initial costing of the query) is executed using its own run-time connection thread, which is returned to the pool when the query is completed.

Oracle Data Source Limitations

This section describes limitations of Oracle data sources that use an OCI or thin driver.

- In a MERGE statement, you cannot modify columns that are also in the ON clause. For example, the column vendor_name cannot be updated:

```
MERGE INTO output_table USING vendor_list ON vendor_name = Widgets
UPDDATE SET vendor_name = V9187
```

- TDV never pushes a MERGE statement with a DELETE branch, because Oracle's behavior is not ANSI-compliant. (Oracle can only delete a row if the row is first updated and then meets certain conditions.)
- To see what JDBC drivers support various versions of Oracle databases, search the Oracle web site.
- Inconsistent capitalization of projection names in query text can result in an Oracle Invalid Identifier error.

Note: Oracle supports all ANSI-supported joins including LEFT OUTER JOIN and RIGHT OUTER JOIN.

SAP HANA Data Source Limitations and Characteristics

These notes pertain to SAP HANA SPS 09:

- SAP HANA DATETIME types have an ABAP-specific special value of EMPTY, which is not the same as NULL. The SAP HANA JDBC driver passes both EMPTY and NULL values to TDV as NULL, and so the results of some calculations made in TDV versus in SAP HANA may differ.

  For example, the function ADD_DAYS (DATE_ADD in TDV), when given an EMPTY value and 1 as arguments, returns 0001-01-02 when executed in SAP HANA, and NULL when executed in TDV, because TDV sees the first argument as NULL.

- SAP HANA supports the return of cursors from procedures. You need to expand the cursor signature and save that change on the procedure; otherwise, the procedures will not display the results. You can check Design Mode in Studio and then use Design by Example to do this. See Defining a Cursor for Projection, page 300, and Designing a Cursor by Example, page 301.

- SAP HANA analytical views are multidimensional objects similar to OLAP cubes. A simple “SELECT *” query or “Show Contents” on such a view does not work. A valid query against an analytical view must contain an aggregate function and a GROUP BY clause.

- The following TDV functionality cannot be used with SAP HANA analytical views due to their multidimensional nature:

  — Caching

  — Statistics gathering

  — Discovery

Sybase Data Source Characteristics

These notes pertain to Sybase SQL support:

- Different versions of Sybase and Sybase IQ handle SQL differently. Refer to Sybase IQ documentation for more information.

- A packaged query should be used if Sybase-specific SQL is required for a given case. TDV supports a superset of SQL-92, but you can force the use of Sybase-native SQL by creating a packaged query containing the Sybase SQL code. TDV does not evaluate SQL inside of packaged queries.

- Because Sybase IQ does not count trailing spaces in its LENGTH function, it might be best to set the TDV configuration parameter to Ignore Trailing Spaces.
- The Sybase data type UNSIGNED BIGINT is not fully supported by TDV queries when its value is larger than +9,223,372,036,854,775,807, because of the limitations of the TDV JDBC type LONG.MAX_VALUE.
- Queries submitted to Sybase IQ cannot contain column names in the IN clause. The IN clause can contain constants only.
- ORDER BY is not allowed in a Sybase IQ subquery. (See Sybase IQ documentation.)

Teradata Data Source Introspection

Introspection of Teradata data sources has these characteristics:

- Introspection of a Teradata RDBMS data sources reveals tables, views, procedures, functions, and macros.
- Introspected Teradata functions and macros are displayed as procedures in the Studio resource tree.
- Teradata macro parameters map to TDV parameters, with the appropriate TDV data types and references for inputs and results.

Vertica Data Source Limitations

This section lists the general limitations related to Vertica. Other limitations depend on the type of features you want to use with Vertica data sources.

- Vertica has no user-defined indexes. Studio cannot index Vertica data sources.
- You cannot use Boolean expressions in a projection. Rewrite the view or use a packaged query.

Netezza Setup

The TDV Server can push Netezza SQL analytic functions, aggregate functions, regular expressions, and associated keywords. For more information, see the *TDV Reference Guide*.

To introspect data, TDV requires SELECT permissions on the following Netezza tables and views.

System Tables and Views	<CurrentCatalog>.<CurrentSchema> Tables and Views
• ADMIN._T_AGGREGATE • ADMIN._T_PROC	• _v_function • _v_aggregate • _V_JDBC_PROCEDURE_COLUMNS2 • _V_JDBC_PKFK2 • _V_JDBC_PROCEDURES2 • _V_JDBC_PRIMARYKEYS2 • _V_JDBC_INDEXINFO2 • _V_JDBC_COLUMNS2 • _V_JDBC_TABLES2 • _V_JDBC_PROCEDURES2

After successful connection with the data source, Netezza tables, views, user-defined aggregates (UDAs), and user-defined functions (UDFs) are displayed for TDV introspection.

Adding Relational Data Sources

The process of adding relational data sources for use with TDV is fundamentally the same for all types.

In the New Physical Data Source dialog, you need to provide the connection details that are used each time TDV accesses the data source. You can edit this information later if necessary when you open this data source in Studio. TDV automatically populates some fields for each data source type.

For descriptions of the fields on the two data source configuration tabs, go to:

- Basic Settings for a Relational Data Source, page 87
- Advanced Settings for a Relational Data Source, page 93

To add a relation data source

1. Make sure that any required drivers are installed and configured for use with Studio. For details on obtaining the adapter and for proper placement of the JDBC JAR files, see the *TDV Administration Guide*.
2. For Netezza, go to Administration > Configuration and set the "Check if nested aggregates is supported by datasource" configuration parameter to True if that is the case.
3. For Hive 0.10, create a view of the table and use Show Contents from the view.
4. Right-click at an appropriate location in the resource tree, and select New Data Source.

   Studio displays the first frame of the New Physical Data Source dialog, which lists the available data source adapters, including custom adapters that may have been configured and added to TDV.
5. Search for and select the data source adapter type, and click Next.
   - For Cloudera CDH4, select Hive 0.10 (HiveServer2) as the data source adapter.
   - For DB2 v9.5, select one of the DB2 v9 adapters.
   - For Microsoft Access data sources on non-windows platforms, select the Non ODBC Microsoft Access adapter.
   - For Oracle 11g using the type 2 native client (OCI), select Oracle 11g (OCI Driver) as the data source adapter.
   - For Oracle 11g using the Type 4 Java-only client (thin driver), select Oracle 11g (Thin Driver) as the data source adapter.
   - Select the Sybase option that matches your version of the data source adapter.

If You Are Using Sybase Version	Use Sybase Option	Driver Required
Sybase v12.x	Sybase 12	jConnect for JDBC 16
Sybase v15.x	Sybase 15	jConnect for JDBC 16
Sybase IQ v15 using JDBC Type 4 driver	Sybase IQ	jConnect for JDBC 16
Sybase IQ v15 using JDBC Type 2 driver	Sybase IQ (Type 2)	SQL Anywhere Database Client v12.0.1

Note: If none of the data source types are appropriate, see Custom Adapters, page 70 for more information.

6. Enter the name for the data source as you want it to appear in Studio resource tree.
7. Enter the basic connection information for the data source. See Basic Settings for a Relational Data Source, page 87, for a description of the basic settings.
8. To enable Kerberos security through keytab files, select the Kerberos option on the Basic tab. You can then type values for the Keytab File and the Service Principal Name fields.
9. Enter the connection values and related fields. See Advanced Settings for a Relational Data Source, page 93, for an overall description of the advanced settings.

   Note: Many of these fields control connection pools. Connection pool configuration uses the standard settings that you can find on the internet.
10. Scroll down to make sure that you have specified all necessary information.

    For additional fields and options, see the section about the specific data source you are adding.
11. Click one of these options:
    - Create & Introspect—Introspect the data source immediately. See Introspecting a Data Source, page 192.
    - Create & Close—Save the data source definition but do not introspect now.

Basic Settings for a Relational Data Source

The collection of fields on the Basic tab differs by type of data source. The following table describes the properties for common data sources. For descriptions of properties for advanced data sources, go to Help > Advanced Data Sources from the Studio main menu.

Basic Property	Data Source	Description
Authentication	Various	Select BASIC or Kerberos authentication method, where offered. See the *TDV Administration Guide* for more information about Kerberos authentication.
Authentication Domain	Composite	The domain where TDV is installed.

Basic Property	Data Source	Description
Character Set	Microsoft Access	See Supported Character Encoding Types, page 165.
	Various DB2	ASCII or EBCDIC.
Create tables with this number of partitions...	SAP HANA	The number of partitions you want used when tables are created. This number affects whether and how table partitioning is done when TDV creates new tables in SAP HANA, for example as cache targets. • If you specify a positive number x (3 to 5 recommended per SAP HANA node), the CREATE TABLE DDL will contain PARTITION BY ROUNDROBIN PARTITIONS x. • If you specify zero, the CREATE TABLE DDL will not include a PARTITION BY clause. Ultimately, the number of partitions affects performance when querying the resulting table, so you should optimize it for your SAP HANA instance and usage.
Complete connection URL string	Apache Drill	A URL to connect to the physical data source. TDV does not validate modifications. The data source adapter might not validate changes. jdbc:drill:drillbit=<hostIP>;schema=<scemaname>
Database	DataDirect Mainframe	For DB2 DBMS types, name of the underlying data source. For other DBMS types, enter NONE.
Database Name	All except Composite and Netezza	Name or alias of the underlying data source. TDV Server uses this name to find and connect to the data source.
	Composite	The name of the published TDV database.
	Netezza	The name of the Netezza catalog. The Netezza catalog is equivalent to a database name and allows view of the databases in the catalog.
Schema	Apache Drill	Name or alias of the underlying data source.

Basic Property	Data Source	Description
DBMS Type	DataDirect Mainframe	DataDirect SQL access keyword that specifies the means by which the adapter accesses the data. Most of the storage schemes listed below do not allow or are not optimized for SQL push. To use more than one DBMS type, create more data source connections in Studio. Supported DBMS type values are: • ADABAS—A very fast transaction processing mainframe database. • DATA—May be used to access both VSAM and sequential files. • DB2—Provides the broadest coverage of SQL-99 functionality, and the most efficient SQL push to the DataDirect Mainframe. Requires a database name. • IMSDB—Provides access to multi-dimensional databases. Appearance of query results differs from conventional relational database results. • VSAM—A read-only storage system. • VSAM/CICS—Transaction processing data system for on-line and batch systems that allows both read and write access.
DSN	Microsoft Access	The Data Source Name. You might need to create a new User or System DSN using the ODBC Data Source Administrator utility (available with Windows Administrative Tools).
Enable SSL	Redshift	
Host Hostname	All	Name or IP address of the machine hosting the data source. Oracle data sources: you cannot enter an Oracle database link name in this field.
Instance Number, if applicable	SAP HANA	Instance number of the SAP HANA system.

Basic Property	Data Source	Description
Login, User, Password	All	User name and password required to access the data source. When the data source is used as a target for cache tables or data ship, the user must also have permission to create tables, execute DDL, and perform other required tasks. Refer to the individual data source descriptions for details.
Net Service Name	Oracle with OCI Driver	The TNS name that is set up through Oracle Net Configuration Assistant.
Pass-through Login	All	Disabled (default)—This allows automated provisioning of a connection pool. Open connection threads can be used by authorized users after the validation query verifies connection status. If pass-through login is disabled, the Save Password check box is not available. Enabled—A new connection to the data source uses the credentials supplied by the client when data is requested from that data source for the first time. Subsequent requests by the same user reuse the existing connection. When another user attempts to connect to a data source, a new connection is created. See "Managing Security for TDV Resources" in the *TDV Administration Guide* for details.
	Oracle	To use Kerberos tokens, Oracle data sources must enable pass-through login. In addition, an enabled Kerberos security module must be running on TDV so that Kerberos-authenticated users with session tokens can use it when submitting queries.
	Sybase	If you choose Kerberos authentication for Sybase, do not enable pass-through login.
Plan	DataDirect Mainframe	Data isolation plan. The default value, SDBC1010, specifies cursor stability. Other values specify repeatable reads, read stability, or uncommitted reads. See the *Shadow RTE Client/adapters Installation and Administration Guide* for more information.

Basic Property	Data Source	Description
Port	All	Port number for the data source to connect with the host. The default port number can depend on data source type. For example: Composite—9401DataDirect Mainframe—1200 DB2—no defaultDB2 z/OS—446 Greenplum—5432HBase—2181 Hive—10000Informix—1526 Microsoft SQL Server—1433MySQL—3306* NeoView—18650Netezza—5480 Oracle—1521PostgreSQL—5432 SAP HANA —30015Sybase—4100** Sybase ASE—5000**Sybase IQ—2638** Teradata—1025Vertica—5433 * Default is 3306 or the base port setting plus eight, depending on the MySQL instance. ** The port number used for communicating with TDV is the TCP/IP port set in the <Sybase_name>.cfg configuration file.
Save Password (check box)	All	By default, the login and password are saved to create a reusable TDV Server system connection pool, usable only by the resource owner, explicitly authorized groups and users, and TDV administrators. They can: • Introspect or reintrospect the current data source • Add or remove data source resources • Perform queries, updates, and inserts on tables • Invoke a stored procedure • Refresh a cached view based on data source resources • Use the query optimizer to gather statistics To disable saving of the password, enable Pass-through Login.
Server	SAP HANA	Name or IP address of the machine hosting the data source.
Service Name	Oracle	Name of the database service.

Basic Property	Data Source	Description
Service Principal Name	DB2 and DB2 z/OS Sybase MS SQL Server Greenplum	This field is available only if you choose Kerberos authentication.
Kerberos Server Name	Greenplum	This field is available only if you choose Kerberos authentication.
Include Realm	Greenplum	This field is available only if you choose Kerberos authentication.
Keytab File	MS SQL Server Greenplum	This field is available only if you choose Kerberos authentication. Use to enable Kerberos security through keytab files. Type the full path to the keytab file.
Ticket Cache (Oracle Thin Driver with 11g data source)	Oracle	Specify when using Kerberos authentication.
Transaction Isolation	All	The degree to which transactions are isolated from data modifications made by other transactions. Netezza and Oracle have only Read Committed (default) and Serializable. Read Uncommitted—Dirty reads, nonrepeatable reads, and phantom reads can occur. Read Committed—Nonrepeatable reads and phantom reads can occur. Repeatable Read—Only phantom reads can occur. Serializable—Dirty reads, nonrepeatable reads, and phantom reads are prevented. None

Advanced Settings for a Relational Data Source

The collection of fields on the Advanced tab differs by type of data source. The following table describes the properties for common data sources. For descriptions of properties for advanced data sources, go to Help > Adapter Help from the Studio main menu.

Advanced Property	Data Source	Description
Collation Sensitive	All	TDV does not use the SORT MERGE join algorithm if any data source involved in the join is marked Collation Sensitive.
Concurrent Request Limit	All	Works with the Massively Parallel Processing engine configuration parameters to control the amount of parallelization for the queries for a particular data source.

Connection Check-out Procedure
 	
Connection checkout procedure
	All	A procedure that returns a valid SQL statement that can be used to initialize the connection—for example, an Oracle Virtual Private Database (VPD) system.
		
		VPD is a method of doing row-level security. After the connection is made, often with a generic account, the client enables certain sets of access rights by setting a security context. In this case, the init procedure returns something like dbms_session.set_identifier ('username'). This is executed on the connection, changing connection privileges from the default to those associated with the user.
		
		Other parameters can also be changed. For example (Oracle), a block like this might be returned by the init procedure:
		`BEGIN`
		`dbms_session.set_identifier('username');`
		`EXECUTE IMMEDIATE 'alter session set optimizer_index_cost_adj=10';`
		`EXECUTE IMMEDIATE 'alter session set optimizer_index_caching=90';`
		`EXECUTE IMMEDIATE 'alter session set "_complex_view_merging"=true';`
		`END;`
		
		The signature of the init procedure should look like this:
		`(IN ds_name VARCHAR, OUT sqlText VARCHAR)`
		
		Write the code in such a way that the init procedure revokes rights if it is not called within the appropriate context.

Advanced Property	Data Source	Description
Connection checkout timeout	Apache Drill	Time that a connection doing a checkout can remain idle without being dropped.
Connection Pool Idle Timeout	All	Number of seconds (default 30) that a connection can remain idle without being dropped from the pool when there are more than the minimum number of connections.
Connection Pool Maximum Size	All	Maximum number of connections (both active and idle) allowed for the data source. When the maximum is reached, new requests must wait until a connection is available. If the maximum number of connections is in use when a request comes in (even with pass-through authentication), the new request is blocked and queued until a connection is available or the Connection Pool Idle Timeout is reached. If no connection was made available within the specified timeout, a check is made for an available connection by the same user. If none is available, the least recently used connection for another user is dropped and a new connection is opened. Studio reuses pooled connections if they continue to be valid after changes (such as connection name), but JDBC requests are forced to use new connections if any part of the data source connection configuration has changed.
Connection Pool Minimum Size	All	Minimum number of connections in the pool even when the pool is inactive. When a connection has been idle, a validation query is used to verify whether an open connection is still valid just prior to submission of a request. If the connection is invalid, the connection is discarded and another is used.

Advanced Property	Data Source	Description
Connection Properties—JDBC Connection Properties (button)	All	Lets you specify property-value pairs to pass to the JDBC data source. Click to add custom connection properties for any JDBC data source. Commonly used properties are populated with default values. Use the Add Argument button to specify other properties and values. TDV does not validate property names. Some data source adapters ignore invalid property names or values; others return an error. The driver properties specify connection timeout settings required by specific drivers. To avoid leaving connections open indefinitely, specify properties explicitly for your data source.

Connection Attributes	Apache Drill	Lets you specify property-value pairs to pass to the data source. For example:
		`bootPassword=key attribute`
		`collation=collation attribute`
		`dataEncryption=true attribute`
		`drop=true attribute`
		`encryptionKey=key attribute`
		`encryptionProvider=providerName attribute`
		`encryptionAlgorithm=algorithm attribute`
		`failover=true attribute`

Advanced Property	Data Source	Description
Connection URL Pattern	All	A template for generating a URL to connect to the physical data source. TDV does not validate modifications. The data source adapter might not validate changes.
	DataDirect	Keywords are: • APNA—Application name for SQL grouping. • CPFX—Catalog prefix for DB2 or non-DB2 resources. • DBTY—DBMS type. • HOST and PORT—Network name or IP address, and port number. • SUBSYS—DB2 database names. • TRLT—Turns off truncation for strings greater than 20 characters. • DTFM—Date format to standard ODBC/ISO format: yyyy-mm-dd
	Teradata	Sends debug log messages to system.out rather than to a file. The pattern is jdbc:teradata://<HOST>/DBS_PORT=<PORT>/DATABASE=<DATABASE_NAME>/CHARSET=UTF8,COMPAT_DBS=true. To view debug messages, append ,LOG=DEBUG (including the initial comma) to the pattern.
Connection URL String	All	The URL string generated from the connection URL pattern with the connection information you provide. This string is used by the JDBC adapter to connect to the physical data source. This field cannot be edited. For details, see the section "Connecting through JDBC Adapters" in the *TDV Administration Guide*.

Advanced Property	Data Source	Description
Connection Validation Query	All	A data-source-specific query that the TDV query engine sends to see if the data source connection is valid. This query is executed every time a connection is checked out from the pool. Enter a query that returns quickly. If this query returns a non-error result, the data source connection is considered valid. If this query fails, the connection is discarded and a new connection is checked out from the available pool. No one SELECT statement works with all data sources. To verify that TDV is running and that it can connect to the data source, devise a query against a published table from that data source.
Data source driver doesn't support query timeout	Apache Drill	Select or clear the check box. If cleared, specify an Execution timeout value in seconds
Enable Bulk Load	Various	Related to the data ship feature capability. Several fields are available only if others are checked. For details, see Data Ship Performance Optimization, page 597.
Enable Bulk Import/Export	SQL Server	Related to the data ship feature capability. Several fields are available only if others are checked. For details, see Data Ship Performance Optimization, page 597.
Enable Export To Another Vertica Database	Vertica	Several of these fields are available only if others are checked. For details, see Data Ship Performance Optimization, page 597. For setting up caching using bulk load features, see TDV Caching, page 465. **Note:** All Netezza data sources should be configured to act as data ship targets.

Advanced Property	Data Source	Description
Data Ship • Enable Sybase IQ SQL Location • Sql Anywhere Data Source • SQL Location List • Sybase iAnywhere JDBC Driver	Sybase IQ only	Locations to improve performance if you plan to use this data source for data ship optimization. You can add one or more Sybase locations by specifying the location name and path of the data source for each link. For further information, refer to Configuring Data Ship for Sybase IQ Targets with Location, page 618.
Select mode	Microsoft SQL Server	Direct or Cursor Direct—Sends all results to the adapter in one request. Each statement establishes its own connection to the database using the same connection properties as the original connection, with auto-commit enabled. Java Transaction API (JTA) is not supported. Direct mode does not support operations where the adapter creates a second statement internally. A typical exception message is "Cannot start a cloned connection while in manual transaction mode." Cursor—Allows you to work with a smaller set of rows that are returned by the SQL statements.
Enable FastLoad/FastExport for large tables	Teradata	Use the FastLoad or FastExport utility to speed up queries. Cardinality information determines whether to use Fastpath or JDBC default loading for a given query.
Enable Native Data Loading	All	Let the data source use its proprietary functionality to optimize performance. See About Data Source Native Load Performance Options, page 79.
Enable Oracle Database Link	Oracle	Check to improve performance if you plan to use this data source for data caching or data ship optimization. Also add one or more Oracle database links. See Configuring Native Caching for Oracle, page 515, and Data Ship Performance Optimization, page 597.
Enable Pass-Through Prepared Statements	Oracle	For pass-through to work, the prepared statement must call data from only one Oracle database instance. Prepared statements can use data from multiple tables within a single Oracle database instance.

Advanced Property	Data Source	Description
Enable PostgreSql dblink	PostegreSQL	Check if you plan to use this data source for data caching or data ship optimization. Also add one or more PostegreSQL database links.
Execute SELECTs Independently	All	Lets a SELECT statement be executed using a new connection from the connection pool, and committed immediately after completion. INSERT, UPDATE, and DELETE statements are executed using the same connection as part of the transaction.
Execute SELECTs in separate transactions from INSERTs and UPDATEs	Apache Drill	Lets a SELECT statement be executed using a new connection from the connection pool, and committed immediately after completion. INSERT and UPDATE, statements are executed using the same connection as part of the transaction.
Execution Timeout	All	The number of seconds an execution query on the data source can run before being canceled. Zero seconds (the default value) disables execution timeout, allowing processes to run to completion—for example, resource-intensive cache updates scheduled for non-peak processing hours.
FastExport Session Count	Teradata	The number of FastExport sessions to use.
FastLoad Session Count	Teradata	The number of FastLoad sessions to use.
Ignore Procedure Return Parameter	Sybase	Check to suppress the return parameter for stored procedures. By default (unchecked), return parameters are inserted into procedure definitions by the JDBC adapter.
Case sensitivity mismatch between TDV and data source can be ignored	Apache Drill	Check to ignore case mismatches.
Ignore trailing space mismatches between TDV and the data source	Apache Drill	Check to ignore trailing spaces.
Honor trailing spaces for string comparison	Apache Drill	Check to honor trailing spaces.

Advanced Property	Data Source	Description
Case insensitive string comparison	Apache Drill	Ignore case.
Include Invalid Introspection Objects	Oracle	Check to return all objects during introspection, including invalid objects.
Introspect Procedures	Oracle	Checked by default. Ignoring procedures speeds up the initial introspection when only tables are wanted.
Introspect comments	Oracle	During the introspection process, TDV can retrieve table and column level comments and add them to the annotations field for each resource. Introspecting Data Source Table and Column Comment Metadata, page 202.
Introspect Should Use Column Alias	Sybase	Check to return column aliases when introspecting data sources. By default (unchecked), introspection returns column names.
Introspect Using DBA_* Views	Oracle	Oracle maintains multiple metadata views. By default, TDV introspection uses ALL_* views, which list resources for which the user has access to both data and metadata. DBA_* views show all resources in the database regardless of data access permissions. Refer to Oracle documentation for differences and privileges.
Max Source Side Cardinality for Semi Join	All	See the documentation for semijoins and the *TDV Administration Guide* for more information.
Max Source Side of Semi Join to Use OR Syntax	All	See the documentation for semijoins and the *TDV Administration Guide* for more information.
Maximum Connection Lifetime	All	The number of minutes that a connection that was returned to the pool persists if there are more open connections than the minimum pool size. The duration is calculated from connection creation. Default value is 60 minutes. Set a smaller value if the pool is likely to run out of connections. Be sure to add a validation query. Set a larger value if you want the connections to be held for a longer period. Set a value of 0 to keep connections alive indefinitely.

Advanced Property	Data Source	Description
Min Target to Source Ratio for Semi Join	All	Sets the minimum target-to-source ratio of cardinality for semijoins. Refer to the *TDV Administration Guide* for more information.
Query Banding	Teradata	Turns the query banding feature on (At Session Level) or off (Off). Stores query context information in the Teradata session table so it can be recovered after a system reset. Query banding takes effect when the data source is next used. Connection pooling has no effect on query band data.
QueryBand Properties	Teradata	Click QueryBand Properties on the right to open a dialog box in which to specify property-value pairs to store in the session table. Four properties are available by default. For the first three, if the default value (including brackets) appears alone in the value field, it is replaced with the actual value at run time. • TDV_USER. ID of the TDV user, application or report that originated the request from a middle-tiered application. The default value is <TDV_USER>. • DOMAIN. The default value is <DOMAIN>. • SESSION_NAME. The default value is <SESSION_NAME>. • SYS. The default value is TDV. TDV administrator can click Add property to add custom name-value pairs.
Select Mode	Microsoft SQL Server	Choose one of two values: Cursor or Direct. These two values affect how result sets are created and retrieved when a query is executed against the data source. Cursor—Generates a server-side cursor. Rows are fetched from the server in blocks. Use JDBC statement method setFetchSize to control the number of rows fetched per request. Useful for queries that return more data than what can be cached on the client.

Advanced Property	Data Source	Description
Show All Databases	Sybase Microsoft SQL Server	Check to list all databases accessible using these credentials during introspection. If a SQL Server database is off-line for the instance you are attempting to introspect, you might see NPE exceptions in the log file.
Streaming Results Mode	MySQL	If selected (default), streams the result set row by row from MySQL to TDV. If not selected, MySQL does not send results to TDV until all results are gathered. For details, see the README.txt file in MySQL JDBC adapter's docs directory.
String comparison is case-insensitive	Apache Drill	Check for insensitive string comparisons.
String comparison honors trailing spaces	Apache Drill	Check to honor trailing space differences.
Supports Star Schema	All	Check only if this data source supports very large predicates and very large cardinalities for star schema semijoins. Refer to the section Star Schema Semijoin, page 587, for more information.
Transaction Lock Wait Timeout	SAP HANA	The length of time a transaction is to wait on the lock before quitting.
Transaction isolation	Apache Drill	Valid values: none, Read committed, Read uncommitted, Repeatable read, Serializable.
Use global temp space for temp tables	DB2	Option that can be used to improve performance when using this data source with the TDV data ship feature. This option would allow you to manage the temp tables created for the data source like any other temp table that you have defined in your source database.
Use pass-through user's certificate for encryption	Oracle	When pass-through login is configured for use with an Oracle data source, check this box to include the user's certificate in the SSL negotiation (handshake).
Use X Views	Teradata	Returns data only for rows containing information on objects that the requesting user owns, created, has privileges on, or has been granted access through a current or nested role.

Enabling Bulk Loading for SELECT and INSERT Statements

Optionally, enable bulk loading for SELECT and INSERT statements. Netezza, Teradata, and SQL Server require extra steps to use this feature. Other data sources might support this feature, but they require no extra configuration.

To enable bulk loading for SELECT and INSERT statements

1. Open Studio.
2. Select Administration > Configuration.
3. Locate the Enable Bulk Data Loading configuration parameter.
4. Set the value to True.
5. Click Apply.
6. Click OK.

To complete configuration of Teradata bulk loading for SELECT and INSERT statements

1. Right-click the data source name in the Studio resource tree and select Open.
2. Select the Advanced tab.
3. Set values on the Advanced tab for Enable FastLoad/FastExport for large tables, page 98, FastExport Session Count, page 99, and FastExport Session Count, page 99.
4. Save your settings.

Configuring Web-Based Data Sources

This topic describes how to configure Web-based data sources including WSDL (Web Services Definition Language) and REST (Representational State Transfer) data sources.f

About OAuth Configuration for SOAP and REST Data Sources

OAuth is a standard method for obtaining secure authorization from the Web. The OAuth authorization framework enables a third-party application to obtain limited access to an HTTP service. You are expected to be familiar with the OAuth 2.0 Authorization Framework (RFC 6749).

If 5 seconds is not the appropriate time for the data source to wait for a web page execution to occur, you can modify the Default OAuth Page Execution Timeout configuration parameter value.

The OAuth grant-flows between the client and the resource owner are:

- Authorization code grant—OAuth uses an authorization server as an intermediary to obtain an authorization code. The authorization server authenticates the resource owner and obtains authorization. The resource owner's credentials are not shared with the client.
- Implicit grant—This simplified flow is optimized for browser clients using a scripting language. The client is issued an access token directly; the authorization server does not authenticate the client. However, the access token may be exposed to the resource owner or other applications with access to the resource server.

- Resource owner password credentials grant—The resource owner credentials (username and password) can be used directly as an authorization-code grant to obtain an access token. The credentials should only be used when there is a high degree of trust between the resource owner and the client, and when other authorization grant types are not available. With this grant type, the client can use a long-lived access token or refresh token instead of storing the resource owner credentials for future use.
- Client credentials grant—The client credentials (or other forms of client authentication) can be used as an authorization grant when the authorization scope is limited to protected resources either under the control of the client or previously arranged with the authorization server.
- Custom Grant—You specify the JavaScript or other process to use to obtain the access token to connect to the resource. For example, the client can request an access token using a Security Assertion Markup Language (SAML) 2.0 bearer assertion grant type.

About WSDL Bare and Wrapper Parameter Styles

WSDL bare and wrapped parameter styles control the consumption of data that is returned in a response. Typically if you have a parameter where you expect the response values to be small, you can put the parameter in the WSDL header. If, however, you expect there to be significant volumes of data returned in the response for the parameter, then it is best to place the parameter or set of parameters in the body of the WSDL.

TDV conforms to the open source standard for XML-based web services. Oracle provides useful reference content that describes the standards fully. This section attempt to summarize how TDV interprets those standards for using Bare and Wrapped parameters for your WSDL.

Style	Description
WRAPPED	For multiple parameters that use the BODY location. The wrapper element is the child of the SOAP BODY and the parameter elements are children of the wrapper element.
BARE	For exactly one parameter with a location of BODY and its element is the only child of the SOAP BODY. All other parameters of the same direction must be mapped to another location, for example, HEADER.

WRAPPED

A parameter style of wrapped means that all of the input parameters are wrapped into a single element on a request message and that all of the output parameters are wrapped into a single element in the response message. If you set the style to RPC you must use the wrapped parameter style. The wrapped style tells the Web service provider that the root element of the message represents the name of the operation and that children of the root element must map directly to parameters of the operation's signature.

With wrapped all sub-parameters are in the first level, and the client automatically wraps them in another top element.

Wrapped requests contain properties for each in and in/out non-header parameter. The properties for the method return values of each out non-header parameter, and in/out non-header parameter. The order of the properties in the request is the same as the order of parameters in the method signature. The order of the properties in the response is the property corresponding to the return value followed by the properties for the parameters in the same order as the parameters in the method signature.

Most Web services engines use positional parameter binding with wrapper style. If a service signature changes, clients must also change. With the wrapper style, all parameters have to be present in the XML message, even if the element corresponding to the parameter can be defined as optional in the schema.

The wrapped request:

- Must be named the same as the method and the wrapper response bean class must be named the same as the method with a “Response” suffix.
- Can only have one message part in WSDL, which guarantees that the message (method element with parameters) is represented by one XML document with a single schema.
- Must have parameters present in the XML message, even if the element corresponding to the parameter can be defined as optional in the schema.

BARE

A parameter style of BARE means that each parameter is placed into the message body as a child element of the message root. With BARE there is only a top element parameter with sub-parameters inside. This one BARE parameter is sent directly.

Whether a given SOAP message is valid is determined by the WSDL that it conforms to. Web services using the bare style can have multiple parameters. It is only the actual invocation of the web service where a single parameter is passed.

The Java API for XML-based Web Services requires that parameters for bare mapping must meet all the following criteria:

- It must have at most one in or in/out non-header parameter.
- If it has a return type other than void it must have no in/out or out non-header parameters.
- If it has a return type of void it must have at most one in/out or out non-header parameter.

Adding a new optional element does not affect clients, because binding is always name-based. To get around having to have a contract defined by the schema of the web service, you can define all child elements of the wrapper root element as required. Optional elements must be made nillable.

WSDL and SOAP Data Sources

Web Services Description Language (WSDL) is an XML-based language that describes a Web service. A WSDL file specifies how clients can submit inputs to an operation and what kind of output the client can expect in return.

Web services are Web-based applications that dynamically interact with other Web applications using an XML message protocol such as SOAP 1.1 or 1.2. Web services can be "bound" to any message format and protocol, but three bindings are popular: SOAP, HTTP, and MIME. TDV supports the following profiles for SOAP binding: SOAP over HTTP, SOAP over WSIF JMS, and SOAP over TIBCO JMS. The binding profile allows specification of the transport protocol, encoding scheme, and message style.

The *transport protocol* defines what mechanism is used for transporting the request.

- TDV supports HTTP and JMS transport.

The *encoding scheme* defines how the message is encoded for the transport.

- TDV supports the literal encoding scheme, which uses an XML schema as the definition for how data should be encoded.
- TDV also provides limited support for the SOAP encoding scheme as defined in the SOAP specification.
- TDV does not support multi-dimensional or sparse SOAP arrays.

The *message style* refers to how the request itself is structured.

- TDV supports the Document Literal message style for WSDL and SOAP. Document Literal style messages have a single part whose schema defines the message payload.

This section contains the following topics:

Adding a WSDL or SOAP Data Source

This section describes how to add a WSDL or SOAP data source. Redefining existing WSDL data sources as SOAP data sources might increase your success in integrating those data sources with TDV and other client applications.

For more information on adding data sources, see Adding a Data Source, page 68.

See Retrieving Data Source Metadata, page 189 for how to introspect a data source.

To add a WSDL or SOAP data source

1. In Studio, right-click the folder in which you want to add the WSDL or SOAP data source and choose New Data Source.
2. Select WSDL or SOAP as the data source adapter and click Next.
3. For Name, enter a name for your data source.
4. On the Basic tab, provide this information for a data source.

Field	Description
URL	URL to the WSDL or SOAP data source. WSDLs can be available by URL over any accessible network. A locally mapped WSDL can be introspected using a URL format like the following: `file:///Z:/test.wsdl`
Login	Optionally, provide a valid username to the data source.
Password	Optionally, provide a password to the data source.

Field	Description
Save Password	This check box is enabled only if Pass-through Login is enabled. See About Pass-Through Login, page 77 for more information.
Pass-through Login	Disabled—The default setting. This allows automated provisioning of a connection pool. If Pass-through Login is disabled, the Save Password check box is not available. Enabled—A new connection to the data source uses the credentials supplied by the client when data is requested from that data source for the first time. If Pass-through Login is enabled, the Save Password check box becomes available. See About Pass-Through Login, page 77 for more information.
Authentication	Choose the method of authentication for this data source: BASIC, NTLM, or NEGOTIATE, OAuth, Digest. When selecting OAuth as the authentication mode, another tab will display. Authentication for the data source must be designated as OAuth 2.0 when the physical data source was first added.
Domain	For NTLM authentication only, enter the domain. See "Configuring NTLM Authentication" in the *TDV Administration Guide* for more information.
Service Principal Name	For NEGOTIATE authentication with Kerberos only, enter the service principal name. See "Configuring Kerberos Single Sign-On" in the *TDV Administration Guide* for more information.
SAML Header Class	SAML assertions are defined in the headers of a class. Type the class name that owns the SAML header.

5. If the data source requires client authentication, click the Advanced tab. See Client Authentication for Web Data Sources, page 145 for how to configure client authentication.

   If the data source requires OAuth, select the OAuth 2.0 tab. Specify the values appropriate to the OAuth flow you want to use. For examples, see TDV SOAP and REST OAUTH Examples, page 147. (This tab and the fields are available

to edit after creation of the data source.)The following table describes the values the user is to provide on the OAuth 2.0 tab:

Field	Description
OAuth Flow	These OAuth flows: • AUTHORIZATION_CODE • IMPLICIT—Client Secret and Access Token URI are disabled. • CLIENT_CREDENTIALS—Resource Owner Authentication fields are disabled. • RESOURCE_OWNER_PASSWORD_CREDENTIALS—Client Authentication fields are disabled. • CUSTOMIZED—User-specified flow.
Client Identification	Used in the request-body of token requests. A unique string representing the identifier issued to the client during registration. It is exposed to the resource owner. Format: string of printable characters.
Client Secret	Used in the request-body of token requests. Enabled only for AUTHORIZATION_CODE, and OAuth flow. Format: string of printable characters.
Authorization URI	URI to use for establishing trust and obtaining the required client properties.
Access Token URI	URI to use for communicating access and refresh tokens to the client or resource server. Disabled only for IMPLICIT OAuth flow.
Redirect URI	Client's redirection end point, established during client registration or when making an authorization request. Must be an absolute URI, and must not contain a fragment component. The authorization server redirects the resource owner to this end point.
Scope	Authorization scope, which is typically limited to the protected resources under the control of the client or as arranged with the authorization server. Limited scope is necessary for an authorization grant made using client credentials or other forms of client authentication. Format: one or more strings, separated by spaces.
State	A request parameter used in lieu of a complete redirection URI (if that is not available), or to achieve per-request customization

Field	Description
Expiration Time (Sec)	The lifetime of the access token.
Use Refresh Token To Get Access Token	Checking this box enables the use of refresh tokens to obtain access tokens, rather than obtaining them manually.
Login and Password	User credentials with which the client or resource owner registers with the authentication server before initiating the OAuth protocol.
Pass-through Login	Enabled or Disabled
Authentication	BASIC, DIGEST, NTLM, or NEGOTIATE—The usual collection of authentication methods, used for registration with the authentication server.
Domain	Domain to which the client or resource owner belongs; for example, composite.
Service Principal Name	SPN of the client or resource owner.
Access Token Type	Bearer—User of a bearer token does not need to prove possession of cryptographic key material. Simple inclusion of the access token in the request is sufficient. Query—The query string "?access_token=<token>" is appended to the URL. Not available for SOAP data sources.
Access Token	Credentials used to access protected resources, with a specific scope and duration. Usually opaque to the client.
Get Token button	Initiates acquisition of an access token. Proper information must be configured for this request to succeed.
Refresh Token	Credentials used to obtain access tokens when they become invalid or have expired. Intended for use with authorization servers, not resource servers.
Custom Flow	The name of the custom flow for a data source with an OAuth flow of CUSTOMIZED.

Field	Description

Using Processors check box
 	
and editable text field
	Check this box to use processors. The editable text field allows you to enter JavaScript and XML. You can use this field to add JavaScripts that log in automatically or use XML to customize any part of the authorization or access token that does not conform to specifications.
	
	Editable text field
	
	The editable text field underneath the Using Processors check box can be used to type the XML elements necessary to establish authorization and access tokens. For example:
	`<Authorization> <AuthorizationProcessors>`
	`<AuthorizationProcessor>`
	`document.getElementById('email').value='queenbeeza@gmail.com';`
	`document.getElementById('pass').value='jellypassword';`
	`document.getElementById('loginbutton').click();`
	`</AuthorizationProcessor> </AuthorizationProcessors>`
	`</Authorization>`
	`<AccessToken> <RequestMsgStyle>QUERY</RequestMsgStyle>`
	`<ResponseMsgStyle>FORM</ResponseMsgStyle>`
	`<ExpireTime>1000</ExpireTime> </AccessToken>`

Sensitive Keyword in JavaScript	Click the plus-sign icon one or more times to add tag-keyword pairs to substitute into the JavaScript specified for a custom authorization flow. The values sent to the server are encrypted, and then replaced with their decrypted values where the tags are found in the JavaScript. These pairs are used for the user name, email ID, password, and other sensitive information. Tag—The name of the tag to find in the JavaScript. Keyword—The encrypted value to decrypt and substitute for the tag in the JavaScript.

6. Click Create & Introspect to initiate introspection.

The Data Source Introspection dialog displays, for the specified WSDL, the available services, ports, and operations. Expand the nodes of the Web service and ports to see the operations within each service.

Enabling Tube Logs for SOAP Data Sources

A tube is a basic SOAP message processing unit. You can determine when to collect log information on those packets of information using the TDV configuration parameters.

To enable tube logs for SOAP data sources

1. Open and log into Studio.
2. From the Administration menu, choose Configuration.
3. Search for or select one of the following parameters (under TDV Server > SOAP Sources):

Parameter	Description
Enable Tube log after executing	Enable logging of SOAP messages or set to ALL for all tubes. Valid values are Terminal, Handler, Validation, MustUnderstand, Monitoring, Addressing, At, Rm, Mc, Security, PacketFilter, Transport, and Customized.
Enable Tube log before executing	Enable logging of SOAP messages or set to ALL for all tubes. Valid values are Terminal, Handler, Validation, MustUnderstand, Monitoring, Addressing, At, Rm, Mc, Security, PacketFilter, Transport, and Customized.

4. Click OK to save your changes and exit the window.

Subscribe to a WSDL Offered as a JMS Data Source

Introspection properties are available for both JMS and HTTP ports.

- For HTTP ports and operations only, you can specify the timeout parameter in milliseconds, or type zero for no timeout, or leave the entry null for an operation where the port setting is to take precedence.
- For JMS ports:
  - The Connector should be specified to ensure functionality of the JMS data source connection pools. JMS connectors should be installed by an

administrator. See “Configuring TDV for Using a JMS Broker” in the *TDV Administration Guide*.

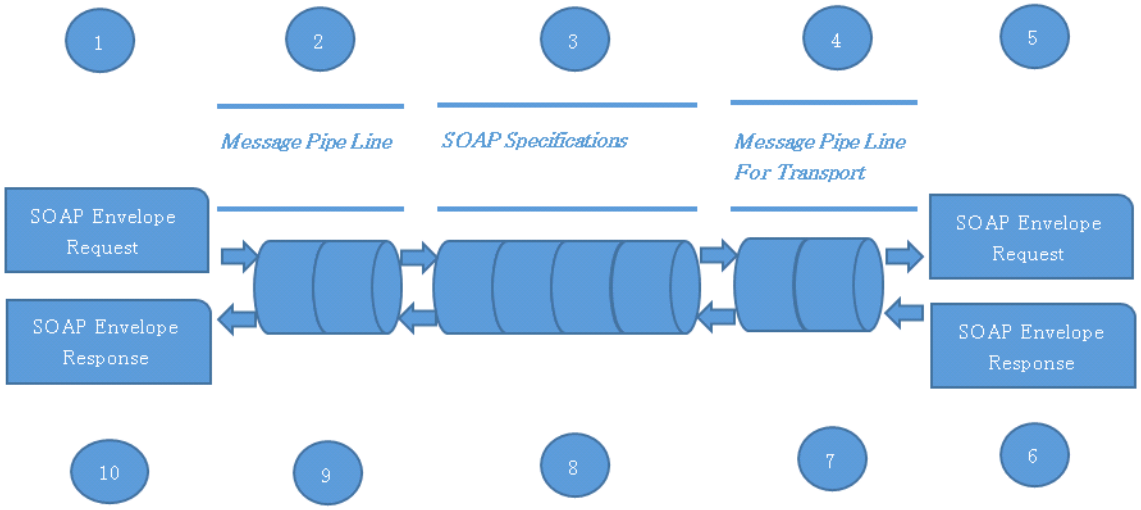
- The specified JMS Destination can be changed to take advantage of different queue destination aliases that offer the same service.
- Delivery Mode can be set to persistent so that messages are written to disk as a safeguard against broker failure. Non-persistent messages are not written to disk before acknowledgment of receipt.
- Message Expiry specifies the period of message validity in the broker queue. An entry of 0 specifies no expiration, while a null entry for an operation specifies that the port setting is to take precedence.
- Operations or messaging priority can be set to an integer of 1 through 9, where 9 is the highest priority.
- Default Timeout is a setting for the consuming client, and it can be set to some duration (in milliseconds). An entry of zero means no timeout, and a null entry specifies that the default takes precedence.
- Individual JMS operations under the port can be configured with a Message Type of Bytes or Text and with specific time-outs tailored to the operation.

If you need to review or change configurations for a Web service, open the Web service from Studio, click the Add/Remove Resources button, make required changes, and reintrospect the data source. For more details about introspection, see Retrieving Data Source Metadata, page 189.

Defining SOAP Message Pipelines

The TDV SOAP data source has a Message Pipeline panel that can be used to configure how request and response messages are handled.

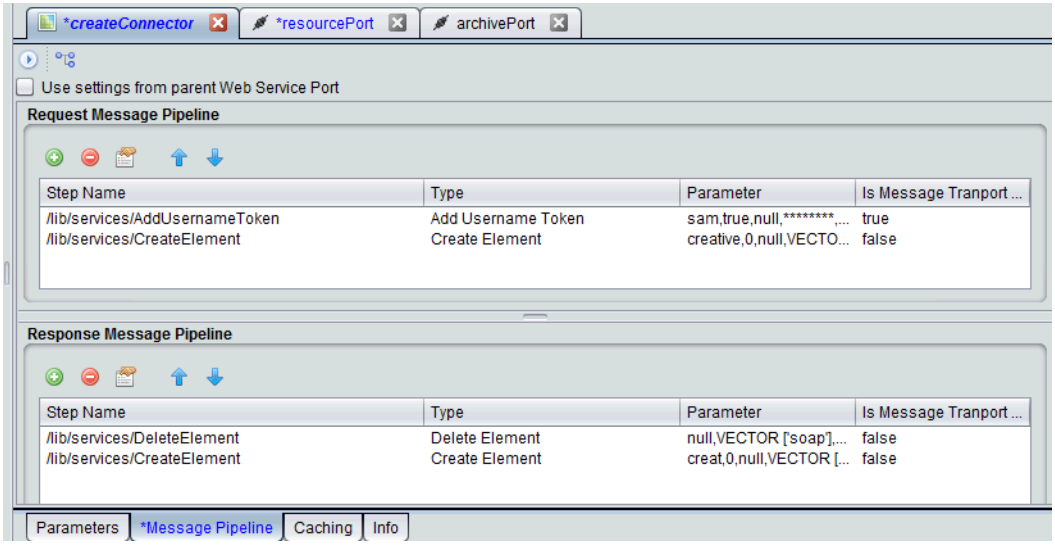
What Happens When a SOAP Message Pipeline is Run



1. Create a SOAP request by executing the operation.	6. The SOAP response message is received from the service side.
2. Message pipelines that are defined as *request*.	7. Message pipelines that are defined as *response*.
3. The SOAP request message is handled according to the SOAP specifications that are defined in the contract.	8. The SOAP response message is handled according to the SOAP specifications that are defined in the contract.
4. Message pipelines that are defined as *request*..	9. Message pipelines that are defined as *response*.
5. The SOAP request message is sent to the service side.	10. The SOAP message is received.

To define a SOAP Message Pipeline

1. After creating and introspecting your SOAP data source, open the editor for the port or operation for which you want to define the message pipeline.
2. Select the Message Pipeline tab.



3. Use the green plus buttons to add steps to the Request Message Pipeline or Response Message Pipeline. The following types can be added:

Request Message Step Types	Response Message Step Types
Add Username Token	Create Element
Create Element	Custom
Custom	Delete Element
Delete Element	Log Message to File
Encrypt Element	Process Security Header
Log Message to File	Set Environment From Node
Set Environment From Node	Set Node From Environment
Set Node From Environment	
Sign Element	

4. Follow the prompts on the screens for what is required for each of the different step types.
5. You can delete, edit, and reorder the steps at anytime after adding them.

Message Level Security (Pipelines) for Legacy Web Services Imported from CAR Files

If messaging passes through intermediate sources in the transport layer (indirect messaging), you must define message-level security. For indirect messaging, or for multiple message receivers or senders, the message needs to be secured at different levels to ensure data integrity.

A pipeline defines multiple instructions that are processed simultaneously. Except for the Custom step, each pipeline step corresponds to a system built-in procedure available at /lib/services/ in the server.

Viewing Message Pipeline Steps

You can provide message-level security by defining multiple steps of well-defined pipeline processing for request-response messages sent through Studio. Specifically, you can provide message-level security at the following locations of a SOAP or WSDL data source:

- SOAP operation

  Operation-level security settings override the security settings of the port.

- WSDL operation

  Operation-level security settings override the security settings of the port.

The message pipeline editor has the following overall characteristics:

- The Request Message Pipeline and Response Message Pipeline sections allow you to add message processing steps to the pipeline.
- You can click the upper Add button to view a drop-down list of the pipeline steps available to process the Request Message Pipeline:

  Create Element, Custom, Delete Element, Log Message To File, Process Security Header, Set Environment From Node, Set Node From Environment
- You can click the lower Add button to view a drop-down list of the available Response Message Pipeline steps:

  Add Username Token, Create Element, Custom, Delete Element, Encrypt Element, Log Message To File, Set Environment From Node, Set Node From Environment, Sign Element
- You can click a Delete button to remove a step.
- You can click an Edit button to change what you specified when you added the step.
- You can change the processing order of a step by selecting it and clicking the Move up or Move down button.

To view the available pipeline steps in Studio

1. Open the operations folder for the Web service that is to be secured.
2. In the editor that opens on the right, the Message Pipeline panel opens.
3. Click the Add button to view the pipeline steps in a drop-down list.

Creating a Log Message to File Pipeline Step

This pipeline step writes SOAP message contents to a file in the specified path. Create this pipeline step for a request message, response message, or both. The outputs from other pipeline steps can be written to the log file created by this step.

This pipeline step corresponds to the system procedure `LogMessageToFile` available in */lib/services/*.

To create the pipeline step named Log Message To File

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline section or the Response Message Pipeline, as needed, and select Log Message To File.

3. In the File Path field, specify a file where the messages are to be logged. The file path is relative to the TDV Server log directory. If the file or the directory does not yet exist, it is created on the local file system of the server on execution.
4. In the File Mode field, specify how the message should be logged.

   APPEND—Adds new messages to the end of the log file.

   OVERWRITE—Causes new messages to overwrite the log file.
5. (optional) Text entered into the Header field is added to the given text in front of the new SOAP envelope message. The text supplies a header note, which is written to the file right before the message contents. This value can be null.
6. (optional) Text entered into the Footer field is added to the end of the processed message. This value can be null.
7. In the Pretty Print drop-down list, select true (default) if you want the message to be formatted with tabbed indents, and false if you do not want the message to be formatted.

Formatting the message can make it easier to read.

8. Click **OK**, and save the step.

The output of this pipeline step is the modified XML document or element. For example:

Adding a Username Token Pipeline Step

You can use this pipeline step to authenticate a response message. If the server hosting the Web service requires that the message has a username token, use this pipeline step.

This pipeline step adds a WS-Security `UsernameToken` to a SOAP envelope. The `UsernameToken` is added to the SOAP header that is identified by the Actor and Must Understand arguments. If the SOAP message does not contain a SOAP header with the specified Actor and Must Understand values, a header is created with those values.

This pipeline step corresponds to the system procedure `AddUsernameToken`, which is available in */lib/services/*.

To create the pipeline step named Add Username Token

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline section, and select Add Username Token.

3. Values for the following fields need to be supplied in the Add Username Token window:

Field	Description of values	Example
Actor	Type a Uniform Resource Identifier (URI) used to specify the recipient of a header element. Specifies the SOAP `Actor` attribute. If it is null, the recipient is the ultimate destination of the SOAP message. The SOAP actor attribute can be used to address the Header element to a particular endpoint. A SOAP message can travel from a sender to a receiver by passing different endpoints along the message path. Not all parts of the SOAP message are intended for the ultimate endpoint of the SOAP message but, instead, are intended for one or more of the endpoints on the message path.	*http://www.w3schools.com/appml* Actor1
Must Understand	Select true or false. Indicates whether a header entry is mandatory or optional for the recipient to process. If you specify Must Understand=true to a child element of the Header element, it indicates that the receiver processing the Header must recognize the element. If the receiver does not recognize the element it must fail when processing the Header. If you specify Must Understand=false to a child element of the Header element, it indicates that the receiver processing the Header need not recognize the element.	true
Username	Valid user name to access the Web service server.	joeuser
Password	Valid password associated with Username.	password
Password Type	Specify the password type, to determine how the password is encoded in the Username Token: DIGEST—Password is rendered in a digested text in the message TEXT—Password is rendered in clear text in the message	DIGEST

4. Click **OK** after supplying all the required information, and save the step.

The `UsernameToken` is added to the SOAP header that is identified by the entry supplied in the Actor field and Must Understand field. If the SOAP message does not contain a SOAP header with the specified Actor and Must Understand values, a header is created with those values.

The following shows sample output:

The digested password is rendered as follows:

```
<wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-user
name-token-profile-1.0#PasswordDigest"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-ws
s-wssecurity-secext-1.0.xsd">APy0l8XZkRsJH7qiAwC6W11D+gs=</wsse:Pa
ssword>
```

Creating an Element Pipeline Step

This pipeline step creates a child element in an XML document or within another element.

This pipeline step corresponds to the system procedure `CreateElement`, which is available in
/lib/services/

To create the pipeline step named Create Element

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline section, and select Create Element.
3. Supply the fully qualified name of the element in the Element Name field.

   This value cannot be null.
4. Specify the position of the element in the Element Position field.

   This position is relative to the element's siblings. The default value is 0 (zero). A value of zero indicates that the element should be created before any existing children. A value of `-1` (negative one) indicates that the element should be created after all existing children.
5. Type the path to the parent of the element in the Parent XPath field.

   The Parent XPath value is used to select the parent element, relative to the root of the element to be created. This entry cannot be null.
6. In the Prefix field, specify the namespace prefix used in the Parent XPath field.
7. In the Namespace field, specify the namespace URIs used in the Parent XPath expression.
8. Optionally, click the Add button next to Declare namespaces used in XPath expressions to define an additional prefix-namespace pair.
9. Optionally, click the Delete button next to a prefix-namespace pair to delete the pair.

10. Click **OK**.

Creating a Pipeline Step to Process a Procedure

This pipeline step processes a custom procedure supplied to it. The signature of a custom procedure supplied to a message pipeline should follow these rules:

- It must not have any OUT parameters.
- It must have only IN or INOUT parameter. There can be more than one IN or INOUT parameter in the signature.
- The first parameter must be of type XML, which can be defined as an IN or INOUT parameter.

The example here uses the following custom procedure (`CustomPipelineStep`) which calls the system procedure `/lib/debug/Log`:

```
PROCEDURE CustomPipelineStep(IN document XML, IN param1 INT)
    BEGIN
        CALL /lib/debug/Log(CAST(document AS VARCHAR(4096)));
    END
```

The first parameter is an input parameter of type XML.

To create the pipeline step named Custom

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline or Response Message Pipeline section, and select Custom.
3. Use the Browse button to locate and select the custom pipeline procedure.
4. Supply the input parameter value in the Parameter Values field.
5. Optionally, select a parameter value and click Edit to change the value.
6. Click **OK**.

   The following shows a sample log message:

Creating a Pipeline Step to Delete an Element

This pipeline step deletes one or more nodes from an XML document or element.

The example used here deletes the Header element named `MyCustomHeader` which was created in the section Creating an Element Pipeline Step, page 122.

This pipeline step corresponds to the system procedure `DeleteElement`, which is available in */lib/services/*.

To delete one or more nodes from an XML document or element

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline or Response Message Pipeline section, and select Delete Element.Type the path to the parent of the element in the Parent XPath field.

   This path is used to select the nodes to be deleted. The path is evaluated against the root node. All resulting nodes are deleted. This entry should not contain any namespaces.
3. Optionally, click the Add button next to Declare namespaces used in XPath expressions to define an additional prefix-namespace pair.
4. Optionally, click the Delete button next to a prefix-namespace pair to delete the pair.
5. In the Prefix field, specify the namespace prefix used in the XPath field.
6. In the Namespace field, specify the namespace URIs used in the XPath field.
7. Click **OK** and save the step.

Creating a Pipeline Step to Encrypt an Element

This pipeline step is used to encrypt the BODY element in the specified SOAP envelope using a symmetric key that is encrypted by a certificate or public key.

This pipeline step corresponds to the system procedure `EncryptElement`, which is available in */lib/services/*.

To create the pipeline step named Encrypt Element

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Response Message Pipeline section, and select Encrypt Element.

UI Element	Description
Procedure path	The path to the pipeline step procedure (by default, */lib/services/EncryptElement*).
Actor	Type a URI. See To create the pipeline step named Add Username Token, page 120.
Must Understand	Select true or false.

UI Element	Description
Element Name	The default value of the Element Name field specifies the schema of the SOAP message that is encrypted. This procedure can be used to encrypt the SOAP message body. The Element Name field can be null, but the default value of `{http://schemas.xmlsoap.org/soap/envelope/}Body` is used to specify the message body for encryption: `{http://schemas.xmlsoap.org/soap/envelope/}Body`
Encryption Algorithm	Accept the default algorithm AES_128 or select a different one if you have installed an unrestricted Java Cryptography Extension (JCE) policy file in the server's JVM. Determines the method of encryption. The default value of AES_128 is sufficient for most purposes. Stronger encryption algorithms such as AES_192 or AES_256 require that an unrestricted Java Cryptography Extension (JCE) policy file be installed in the server's JVM.
Certificate Alias	Select or type an appropriate certificate alias. If the WSDL data source or Web service has a list of certificates associated with it, the aliases of those certificates are listed here. The alias of a certificate or public key in the key store that is used to encrypt the symmetric key that is used to encrypt the element. It cannot be null. Certificates are associated with a data source using the Data Source connection properties. To access the certificates, open the editor for the WSDL data source and click the Advanced tab in the Connection Information section. Any Alias listed in the Certificates table can be used as the Certificate Alias in a pipeline step.

3. Supply all required values and click **OK**.

Creating a Pipeline Step to Process a Security Header

This pipeline step is used to process a WS Security SOAP header in a SOAP envelope, as follows:

- If the envelope contains a WS Security header with the specified actor, the header is processed.
- All security elements in the header are evaluated.
- If any header security element indicates that the envelope contains signed elements, the signatures of those elements are verified.

- If any header security element indicates that the envelope contains encrypted elements, those encrypted elements are decrypted.

This pipeline step corresponds to the system procedure `ProcessSecurityHeader`, which is available in */lib/services/*.

To create the pipeline step named Process Security Header

1. Open the pipeline editor for the resource that is to be secured.
2. Click the Add button in the Request Message Pipeline section, and select Process Security Header.
3. Supply the path to the pipeline step procedure if it is different from */lib/services/ProcessSecurityHeader*.
4. Supply a value in the Actor field, if necessary.

The Actor determines which WS Security header to process. It can be null.

5. Click **OK** to save the step.

Creating a Pipeline Step to Set Environment From Node

This pipeline step saves an element or attribute value as an environment variable. It evaluates the given xpath expression against the envelope, and stores the result in the environment in a variable with the specified name. The result of the xpath expression is interpreted as a single string.

This pipeline step corresponds to the system procedure `SetEnvironmentFromNodeValue`, which is available in */lib/services/*.

To create the pipeline step named Set Environment From Node

1. In the pipeline editor for the resource that is to be secured, click the Add button in the Request Message Pipeline or Response Message Pipeline section and select Set Environment From Node.
2. Type the path and step name in the Procedure Path field if it is different from */lib/services/SetEnvironmentFromNodeValue*.
3. Type the XPath value in the XPath field.

   The XPath entry is evaluated to some text which is stored in the variable name provided in the Variable Name field.
4. Type the variable name in the Variable Name field, without any spaces in the name string.

   Variable Name is the name of an environment variable. It is an arbitrary string, and it is not case-sensitive. (Both `sample` and `SAMPLE` are considered the same.)

5. In the Prefix field, specify the namespace prefix used in the XPath field.
6. In the Namespace field, specify the namespace URIs used in the XPath expression.
7. Optionally, click the Add button next to Declare namespaces used in XPath expressions to define an additional prefix-namespace pair.
8. Optionally, click the Delete button next to a prefix-namespace pair to delete the pair.
9. Click **OK**.

Creating a Pipeline Step to Set Node From Environment

This pipeline step sets an element or attribute value from an environment variable. The given xpath expression is used to select a node from the envelope. The node value is then set to the value of the specified environment variable.

This pipeline step corresponds to the system procedure `SetNodeValueFromEnvironment`, which is available in */lib/services/*.

To create the pipeline step named Set Node From Environment

1. In the pipeline editor for the resource that is to be secured, click the Add button in the Request Message Pipeline or Response Message Pipeline section and select Set Node From Environment.
2. Type the path and step name in the Procedure Path field if it is different from */lib/services/SetNodeValueFromEnvironment*.
3. Type the XPath value in the XPath field.

The XPath expression is evaluated to an element.

4. Type the name of an environment variable in the Variable Name field, without a space in the name string.

   Variable Name is an arbitrary string, and is not case-sensitive. (Both `sample` and `SAMPLE` are considered the same.)
5. In the Attribute Name field, specify the attribute whose value is to be set from the environment variable.
6. In the Prefix field, specify the namespace prefix used in the XPath field.
7. In the Namespace field, specify the namespace URIs used in the XPath expression.
8. Optionally, click the Add button next to Declare namespaces used in XPath expressions to define an additional prefix-namespace pair.

9. Optionally, click the Delete button next to a prefix-namespace pair to delete the pair.
10. Click **OK**, and save the step.

Creating a Pipeline Step to Sign an Element

This pipeline step is the simplified version of `Encrypt Element`, and is used to sign an element in the specified SOAP envelope using a private key.

This pipeline step corresponds to the system procedure `SignElement` which is available in */lib/services/*.

To create the pipeline step named Sign Element

1. In the pipeline editor for the resource that is to be secured, click the Add button in the Response Message Pipeline section and select Sign Element.
2. Type the path and step name in the Procedure Path field if it is different from */lib/services/SignElement*.
3. For details on Actor, Must Understand, Element Name, and Certificate Alias, see To create the pipeline step named Encrypt Element, page 124.
4. Supply all required values and click **OK**.

REST Data Sources

Representational State Transfer (REST) defines a set of architectural principles by which you can design Web services that focus on systems' resources and how these resource states are addressed and transferred over HTTP by client applications written in different languages.

The REST implementation in TDV establishes a one-to-one mapping between these basic operations and HTTP methods.

Operation	HTTP/REST	SQL Equivalent
Create a resource on the server	POST	INSERT
Retrieve a resource	GET	SELECT
Update or change a resource state	PUT	UPDATE
Remove or delete a resource	DELETE	DELETE

When you define a REST data source, you define its URL connection information and its operations which include the input and output parameters. After definition, the REST data source in Studio contains a Web Service Operation for each defined REST operation. The Web Service Operations in Studio are similar to a stored procedure and can be used in the same way. See Procedures, page 267 for more information.

This section contains the following topics:

Adding a REST Data Source

This section describes how to add a REST data source. For further information, see Adding a Data Source, page 68.

About POST and Multi-Part/Form

Multi-Part/Form supports the transfer of multiple binary types of data for POST operations. For example, you could use this to transfer multiple GIF files.

- For POST, the Multi-Part/Form option is enabled for an HTTP 'multipart/form-data' request.
- If the Multi-Part/Form option is chosen, each parameter that uses Body and 'IN' or 'IN/OUT' will be consumed as a string part of the request.
- If the Content-type parameter is defined for an HTTP header then the charset defined in the parameter will be applied to all parts for the request. The default charset is ISO-8859-1.

To add a REST data source

1. In Studio, right-click the folder in which you want to add the REST data source and choose New Data Source.
2. Select REST as the data source adapter and click Next.

   Studio displays the New Physical Data Source dialog for you to define the REST data source.

3. For **Name**, enter a name for your REST data source.
4. On the Basic tab, provide this information for a REST data source:

Field	Description
Base URL	The base URL to access the REST data source. This is in the form: http://<web site name> Example: http://search.twitter.com
Login	Optionally, provide a valid username to access the REST data source.
Password	Optionally, provide a valid password to access the REST data source.
Save Password	This check box is enabled only if Pass-through Login is enabled. See Basic Settings for a Relational Data Source, page 87, for more information.
Pass-through Login	Disabled—The default setting. This allows automated provisioning of a connection pool. If Pass-through Login is disabled, the Save Password check box is not available. Enabled—A new connection to the data source uses the credentials supplied by the client when data is requested from that data source for the first time. If Pass-through Login is enabled, the Save Password check box becomes available. See About Pass-Through Login, page 77, for more information.
Authentication	Choose the method of authentication for this data source: BASIC, NTLM, or NEGOTIATE, OAuth, Digest. When selecting OAuth as the authentication mode, another tab will display. Authentication for the data source must be designated as OAuth 2.0 when the physical data source was first added. For more information, see Client Authentication for Web Data Sources, page 145.
Domain	For NTLM authentication only. Enter the domain name. See "Configuring NTLM Authentication" in the *TDV Administration Guide* for more information.
Service Principal Name	For NEGOTIATE authentication with Kerberos only: enter the service principal name. See "Configuring Kerberos Single Sign-On" in the *TDV Administration Guide* for more information.

Field	Description
Access Token	For OAUTH 2.0 authentication, type the access token.
JSON Format	Check this box if you want to use the JSON (JavaScript Object Notation) standard for data interchange. When this box is checked, each HTTP request-response pair is encoded and decoded in the JSON format.
BadgerFish Enabled	Check this box to enable BadgerFish if the REST services outside of TDV use the BadgerFish convention to translate XML into the JSON format. JSON Format must also be checked.
Primitive Value Format	Check this box to read and send the value in its primitive presentation. JSON Format must also be checked.
Package Name	Prefix for each element to make the service name unique. The package name plays the same role as namespace in the XML format. Package name can consist of any characters that are legal in Java class names. JSON Format must also be checked.
Wrapper of JSON Bare Response	Type in the wrapper name. It can be the wrapper key of the whole JSON response. This value makes it possible to convert JSON responses to well-formed XML value types. JSON Format must also be checked.

5. On the bottom part of the Basic tab, define the REST operations.
   a. Under Operations, click the Add Operation button to add an operation.
   b. In the New Operation dialog box, enter a name for the operation and click OK.
   c. For each operation, you must define an HTTP Verb and Parse an Operation URL.

Field	Description
HTTP Verb	Choose the operation type: GET, POST, PUT, or DELETE.
Operation Name	Automatically filled in with the operation name you entered and selected in the Operations box.

Field	Description
Operation URL	Enter the URL for the REST operation in the format: `<operation_api_name>?<query_parameters>` • <operation_api_name>—The API service operation name as defined by the URL. Examples: addfeed.atom and search.atom from Twitter. You need to know the operation names for your Web data source and follow their conventions. • <query_parameters>—Separate multiple entries with & (ampersand). Can be one or more of the following: • <constant>—Define one or more constants in the form constant=value. Example: count=20 • <{URL_parameter}>—Define one or more URL parameters in curly brackets. Example: tweet={mytweet} At execution time, the Base URL is combined with the Operation URL defined for each operation to form the data request.
Parse	Click Parse to add all URL parameters specified in curly brackets in the Operation URL to the URL Parameters list. If the parser finds the syntax correct, it adds it to the URL Parameters list. For example, if you enter tweet={mytweet} in the Operation URL field and click Parse, mytweet appears under Param Name in the URL Parameters section.

d. For Request/Response Style, select one of the following:

Radio Button	Description
Bare	To support one parameter.
Wrapped	To support multiple parameters within a named wrapper. For Wrapped only, enter: • Request Wrapper QName—Specify a unique name for the parent element that is to contain the input parameters that you are wrapping. • Response Wrapper QName—Specify a unique name for the parent element that is to contain the output parameters that you are wrapping.
Multi-Part/Form	To support the transfer of multiple binary types of data for POST operations. For example, you could use this to transfer multiple GIF files.

e. For URL parameters, you can edit parameters and their data types for those parameters that you are passing through the operation URL.

f. For Header/Body Parameters, define the header and body parameters and their information. If you want to use the design by example feature, you must save your data source definition to activate the Design By Example button. For more information on how to use this feature, see Using Design By Example to Infer Parameter Data Types, page 139.

— Click Add Operation to add a parameter. A new row appears in the Header/Body Parameters section with no name but with a default location (Body), data type (VARCHAR) and in/out direction (IN).

— Double-click under Param Name and type the name of the parameter.

— Click under Location, and from the drop-down list choose one of these options:

HTTP Header—Parameter is located in the HTTP header.

Body—Parameter is located in the body of the XML file.

— Click under Data Type, and from the drop-down list choose one of these options:

Option	Description
Decimal	DECIMAL, DOUBLE, FLOAT, or NUMERIC.
Integer	BIGINT, BIT, INTEGER, SMALLINT, or TINYINT.
String	CHAR, CLOB, LONGVARCHAR, or VARCHAR.
Time	DATE, TIME, or TIMESTAMP.
Complex	< >XML Complex XML is any XML defined outside of the W3C XML standard. If your XML has customized elements or requires and XSD file to interpret the meaning of the XML elements within it, the XML is considered complex.
Browse	Choose this to browse for an XSD file that defines the data type being used.

— Click under In/Out, and from the drop-down list choose IN for input parameters or OUT for output parameters.

6. If the data source requires client authentication, click the Advanced tab. See Client Authentication for Web Data Sources, page 145 for how to configure client authentication.
7. If the data source requires OAuth, select the OAuth 2.0 tab. Specify the values appropriate to the OAuth flow you want to use.

   The following table describes the values the user is to provide on the OAuth 2.0 tab:

Field	Description
OAuth Flow	These OAuth flows: • AUTHORIZATION_CODE • IMPLICIT—Client Secret and Access Token URI are disabled. • CLIENT_CREDENTIALS—Resource Owner Authentication fields are disabled. • RESOURCE_OWNER_PASSWORD_CREDENTIALS—Client Authentication fields are disabled. • CUSTOMIZED—User-specified flow.
Client Identification	Used in the request-body of token requests. A unique string representing the identifier issued to the client during registration. It is exposed to the resource owner. Format: string of printable characters.
Client Secret	Used in the request-body of token requests. Enabled only for AUTHORIZATION_CODE, and OAuth flow. Format: string of printable characters.
Authorization URI	URI to use for establishing trust and obtaining the required client properties.
Access Token URI	URI to use for communicating access and refresh tokens to the client or resource server. Disabled only for IMPLICIT OAuth flow.
Redirect URI	Client's redirection end point, established during client registration or when making an authorization request. Must be an absolute URI, and must not contain a fragment component. The authorization server redirects the resource owner to this end point.

Field	Description
Scope	Authorization scope, which is typically limited to the protected resources under the control of the client or as arranged with the authorization server. Limited scope is necessary for an authorization grant made using client credentials or other forms of client authentication. Format: one or more strings, separated by spaces.
State	A request parameter used in lieu of a complete redirection URI (if that is not available), or to achieve per-request customization
Expiration Time (Sec)	The lifetime of the access token.
Use Refresh Token To Get Access Token	Checking this box enables the use of refresh tokens to obtain access tokens, rather than obtaining them manually.
Login and Password	User credentials with which the client or resource owner registers with the authentication server before initiating the OAuth protocol.
Pass-through Login	Enabled or Disabled
Authentication	BASIC, DIGEST, NTLM, or NEGOTIATE—The usual collection of authentication methods, used for registration with the authentication server.
Domain	Domain to which the client or resource owner belongs; for example, composite.
Service Principal Name	SPN of the client or resource owner.
Access Token Type	Bearer—User of a bearer token does not need to prove possession of cryptographic key material. Simple inclusion of the access token in the request is sufficient. Query—The query string "?access_token=<token>" is appended to the URL. Not available for SOAP data sources.
Access Token	Credentials used to access protected resources, with a specific scope and duration. Usually opaque to the client.
Get Token button	Initiates acquisition of an access token. Proper information must be configured for this request to succeed.
Refresh Token	Credentials used to obtain access tokens when they become invalid or have expired. Intended for use with authorization servers, not resource servers.

Field	Description
Custom Flow	The name of the custom flow for a data source with an OAuth flow of CUSTOMIZED.

Using Processors check box
 	
and editable text field
	Check this box to use processors. The editable text field allows you to enter JavaScript and XML. You can use this field to add JavaScripts that log in automatically or use XML to customize any part of the authorization or access token that does not conform to specifications.
	
	Editable text field
	
	The editable text field underneath the Using Processors check box can be used to type the XML elements necessary to establish authorization and access tokens. For example:
	`<Authorization> <AuthorizationProcessors>`
	`<AuthorizationProcessor>`
	`document.getElementById('email').value='queenbeeza@gmail.com';`
	`document.getElementById('pass').value='jellypassword';`
	`document.getElementById('loginbutton').click();`
	`</AuthorizationProcessor> </AuthorizationProcessors>`
	`</Authorization>`
	`<AccessToken> <RequestMsgStyle>QUERY</RequestMsgStyle>`
	`<ResponseMsgStyle>FORM</ResponseMsgStyle>`
	`<ExpireTime>1000</ExpireTime> </AccessToken>`

Sensitive Keyword in JavaScript	Click the plus-sign icon one or more times to add tag-keyword pairs to substitute into the JavaScript specified for a custom authorization flow. The values sent to the server are encrypted, and then replaced with their decrypted values where the tags are found in the JavaScript. These pairs are used for the user name, email ID, password, and other sensitive information. Tag—The name of the tag to find in the JavaScript. Keyword—The encrypted value to decrypt and substitute for the tag in the JavaScript.

8. Click Create & Close to save the REST data source.

Studio adds the REST data source to the resource tree and opens the Data Source Introspection window for you to introspect the new REST resource.

Setting a Value for NULL JSON Values in REST Responses

There is a configuration parameter that you can use to set the value of a JSON null the occurs in a REST response. The default value for the parameter is "json_key": NULL. If that value is acceptable, there is nothing to do. If you would like to customize that value, you can use the following steps.

To set a value for NULL JSON values in REST response

1. Within Studio, select Administration > Configuration.
2. In the Configuration window, navigate to JSON NULL Value in REST Response.
3. Accept the value of null or type a value that you want to assign to all NULL JSON values in a REST response.
4. Click OK.

Passing a Full XML Document in a POST Request to a REST Service

If your REST data source is configured to accept a set of scalar parameters and inputs, being able to pass the whole XML document to the service might be required. When passing an XML document in a POST request to a REST service, if the parameter name is “[rawdata]” then the submitted data is not wrapped but submitted as-is (and not wrapped as a value for an XML element.

To interface CDVP with a REST Service

1. Define your REST data source.
2. On the Basic tab, scroll down to the Operations section.
3. Define the details for your operation, including the HTTP Verb, Operation Name, and Operation URL.
4. In the Header/Body Parameters table, add an OUT Param Name named response with a Data Type of XML. This is a Body parameter with the direction of OUT.
5. In the Header/Body Parameters table, add an IN Param Name named [rawdata] with a Data Type of XML.This is a Body parameter with the direction of IN.
6. Make sure that the XML <-> JSON check boxes are clear for all the parameters.
7. Save your data source.
8. Invoke the service and provide a full request document as the input parameter.

Using an HTTP Header Parameter to Specify the Content Type

It can be a useful to use a Content-Type HTTP header parameter especially where the REST service requires an XML request but returns a JSON response. When the JSON Format is selected, the REST data source adapter assumes that the format of the request and the format of the response will match. The REST data source automatically generates a header “Content-Type: application/json” in the request. This must be overridden with the value of "application/xml" so that the service can correctly interpret the request.

To use an HTTP header parameter to submit an XML request and get a JSON response

1. Define your REST data source.
2. On the Basic tab, select the check box next to the JSON Format label.
3. On the Basics tab, scroll down to the Operations section.
4. Define the details for your operation, including the HTTP Verb, Operation Name, and Operation URL.
5. In the Header/Body Parameters table, add an OUT Param Name named response with a Data Type of XML. This is a Body parameter with the direction of OUT.
6. In the Header/Body Parameters table, add an IN Param Name named [rawdata] with a Data Type of XML. This is a Body parameter with the direction of IN.
7. Inn the Header/Body Parameters table, add an IN Param Name named Content-Type with a Data Type of string. This is a Header parameter with the direction of IN.
8. Clear all the XML <-> JSON check boxes for the [rawdata] parameter.
9. Select the XML <-> JSON check boxes for the response parameter.
10. Save your data source.
11. Invoke the service and provide a full request document for the [rawdata] input parameter, and providing the string application or XML for the Content-Type parameter.

Using Design By Example to Infer Parameter Data Types

Design by example is a feature that provides automated inference of the definition set (XML schema) based on the response from the REST data sources. After you create your REST data source, you can open the data source editor and infer the data types of the parameters by using the Design By Example button, and then you can edit the type if necessary. Typically, design by example is used to infer the data type of body parameters.

There are several ways to access the design by example functionality, including:

- Creating a Definition Set by Example Using the Data Source Editor, page 139
- Creating a Definition Set by Example Using a Web Service Operation Editor, page 141

Creating a Definition Set by Example Using the Data Source Editor

Whether you are creating a new REST data source or editing an existing one, the way you interact with the editor to gain access to the design by example feature is the same.

To create a definition set by example from the Data Source editor

1. Make sure that all the input parameters defined for your REST data source are defined according to REST operation requirements.

   Because the definition set is created from the REST service response, correct input is important, so that the response is correct.

2. Create your REST data source following the instructions in Adding a REST Data Source, page 129.
3. Make sure that the following is true for the data source:
   - Operations are listed.
   - Operations have been assigned HTTP verbs.
   - If the Operation URL includes a variable, it is in curly brackets ({}). The URL parameters must be parsed and listed in the URL Parameters portion of the screen.
   - It has been saved and introspected.
4. Open the REST data source editor if it is not already open.
5. Select the Configuration tab.
6. Scroll down to the Connection Information portion of the tab and select the Basic tab.

7. Scroll down to the Operations portion of the screen.
8. Under the Header/Body Parameters portion of the screen, click Design By Example.

   If an XML schema file was previously created or specified for this data source, a message pops up. Click OK to continue and overwrite that file with the new information.
9. If your operation has input parameters, type values for the URL Parameters in the Input Values for <op_variable> window. The value that you type must be consistent with the data type of the URL parameter.

   The Add Definition Type dialog appears.
10. In the tree pane of the screen, navigate to a sample definition set. TDV uses the sample definition set to automatically create a new definition set from the JSON or XML responses for your REST operation. After the definition set has been created, you can edit it.

    Typically, design by example logic is used to suggest metadata for body parameter types, which are usually OUT parameters.
11. In the right-side pane, select an Element or Type from the definition set for the Show field, and then select one of the values in the list in the other portion of the screen.

    After you make a selection, the full Studio tree path to that sample definition appears in the Type field.
12. Click OK.

    A new row appears in the Header/Body Parameters section with no name, but with default location (Body), data type, and in/out direction (IN). A new definition set resource is created in Studio under the REST data source with the name of the operation TDV used to suggest the schema.
13. Double-click under Param Name and type the name of the parameter.
14. Validate that the values listed for the other columns are consistent with your design.
15. For JSON formatted data, make your XML <-> JSON choice. Select the check box to convert the result set to XML. Clear the check box to retain the JSON formatting.
16. Save your edited data source.

Creating a Definition Set by Example Using a Web Service Operation Editor

To create a definition set by example from a Web service operation editor

1. Make sure that all the input parameters defined for your REST data source are defined according to REST operation requirements.

   Because the definition set will be created from the REST service response, only correct input of the request could get correct response back.

2. Create your REST data source following the instructions in Adding a REST Data Source, page 129.

3. Make sure that the following is true for the data source:

   — Operations are listed.

   — Operations have been assigned HTTP verbs.

   — If the Operation URL includes a variable, it is in curly brackets ({}), and the URL parameters are parsed and listed in the URL Parameters portion of the screen.

   — It has been saved and introspected.

4. From the Studio resource tree, expand the REST data source node.

5. Select an operation and open its editor.

6. Click Design By Example.

   If an XML schema file was previously created or specified for this data source, a message pops up. Click OK to continue and overwrite that file with the new information.

7. If your operation has input parameters, type values for the URL Parameters in the Input Values for <op_variable> window. The value that you type must be consistent with the data type of the URL parameter.

   The Add Definition Type dialog appears.

8. In the tree pane of the screen, navigate to a sample definition set. TDV uses the sample definition set to automatically create a new definition set from the JSON or XML responses to the REST operation. After the definition set has been created, you can edit it.

   Typically, design by example logic is used to suggest metadata for Body parameter types, which are usually OUT parameters.

9. In the right-side pane, select an Element or Type from the definition set for the Show field, and then select one of the values in the list in the other portion of the screen.

After you make a selection, the full Studio tree path to that sample definition appears in the Type field.

10. Click OK.

    A new row appears in the Header/Body Parameters section with no name, but with default location (Body), data type, and in/out direction (IN). A new definition set resource is created in Studio under the REST data source with the name of the operation TDV used to suggest the schema.

11. Double-click under Param Name and type the name of the parameter.
12. Validate that the values listed for the other columns are consistent with your design.
13. Save your changes.

Cross-Origin Resource Sharing (CORS)

For security reasons, browsers restrict cross-origin HTTP requests initiated within scripts. A web page can typically embed many kinds of content from other domains. However, W3C recommends that cross-origin requests for certain items ("restricted resources") be made only using secure means. Such resources include embedded web fonts and AJAX (XMLHttpRequest) APIs.

Cross-origin resource sharing (CORS) is a mechanism that allows a browser from one domain to request restricted resources from another domain, within well-defined and secure boundaries. TDV supports CORS under Chrome and Firefox browsers, and follows the recommendation for CORS on the Web at http://www.w3.org/TR/cors/.

The administrator of a public REST service in TDV configures the CORS environment as described in this section:

- CORS Configuration, page 143

Typically the client makes two kinds of requests, preflight requests and actual requests:

- Preflight Requests, page 143
- Testing Preflight Requests, page 144
- Actual Requests, page 144
- Testing an Actual Request, page 145

CORS Configuration

The TDV configures CORS using configuration parameters. The parameters are in the Administration > Configuration window under Server > Web Services Interface > CORS. These parameters are described in the following table.

Configuration Parameter	Comments
Allow Credentials	A boolean indicating whether the resource allows requests with credentials. Default value is true, because TDV always allows credentials. When this is false, CORS is disabled.
Allowed Headers	A comma-separated list of HTTP headers that may be specified when accessing the resources. Default value is X-Requested-With, Content-Type, Accept, Origin.
Allowed Methods	A comma-separated list of HTTP methods that can be used when accessing the resources. Default list is GET, POST, HEAD.
Allowed Origins	A comma-separated list of the origins that may access the resources. Default value is * (all origins).
Chain Preflight	If true (default), preflight requests are chained to their target resources for normal handling (that is, handling as OPTION requests). Otherwise, the filter responds to the preflight.
Exposed Headers	A comma-separated list of HTTP headers that may be exposed on the client. Default value is an empty list.
Preflight Max Age	The number of seconds that the client is allowed to cache preflight requests. Default value is 1800 seconds (30 minutes).

Preflight Requests

A preflight request asks the other domain's server which HTTP request methods it supports. This request also asks for an indication of whether cookies or other authentication data should be sent with the actual request.

The preflight request sends its HTTP request using the OPTIONS method, to determine whether the actual request is safe to send. A preflight request is required if the actual request wants to:

- Use a PUT or DELETE method
- Set custom headers in the request—for example, a header such as bb

Testing Preflight Requests

You can debug CORS preflight requests using cURL.

To test a CORS preflight request

1. Send a preflight request using cUrl:

```
curl -H "Origin: http://example.com" \
-H "Access-Control-Request-Method: POST" \
-H "Access-Control-Request-Headers: X-Requested-With" \
-X OPTIONS --verbose \
https://www.googleapis.com/discovery/v1/apis?fields=
```

   This is similar to a regular CORS request, with a few additions:

   — The -H flags send additional preflight request headers to the server.

   — The -X OPTIONS flag indicates that this is an HTTP OPTIONS request.

2. Optionally, use the -H flag to specify additional headers, such as User-Agent.

If the preflight request is successful, the response should include these response headers:

- Access-Control-Allow-Origin
- Access-Control-Allow-Methods
- Access-Control-Allow-Headers

If the preflight request is not successful, one of the following occurs:

- The three response headers listed above do not appear.
- The HTTP response is not 200.

Actual Requests

An actual request includes an HTTP request method and any cookies or other authentication the other domain's server requires. If the actual request qualifies (see list below) as a simple cross-site request, it does not need to be preceded by a preflight request.

A simple cross-site request has the following characteristics:

- It uses only GET, HEAD or POST to send data to the server.
- It does not set custom headers with the HTTP request.
- The data sent to the other domain's server with the HTTP request is of one of these content types:
  - application/x-www-form-urlencoded
  - multipart/form-data
  - text/plain

If an actual request does not qualify as a simple request, it must be preceded by a preflight request. (See Preflight Requests, page 143.)

Testing an Actual Request

You can debug actual CORS requests using cURL, using the URL you are testing in place of the sample Google API, which supports CORS.

To test an actual CORS request

1. Send a regular CORS request using cURL:

```
curl -H "Origin: http://example.com" \
https://www.googleapis.com/discovery/v1/apis?fields=
```

   The -H "Origin: http://example.com" flag is the name of the third-party domain making the request. Substitute the name of your domain.

2. Optionally, use the --verbose flag to print the entire response so you can see the request and response headers.

The response should include the Access-Control-Allow-Origin response header.

Client Authentication for Web Data Sources

When Web data sources require client authentication, a keystore must be specified to identify the TDV Server to the provider. The TDV Server configuration keystore key alias has a default value that names a sample keystore, so that you can use client authentication immediately upon installation.

If the TDV configuration settings for keystore alias are set to null, the method described below to comply with client authentication requirements is used for Web data sources. The TDV configuration to use a specific keystore key alias overrides keystore specification defined on individual data sources.

To specify a keystore to comply with client authentication requirements

1. Open the Web data source in Studio.
2. Click the Advanced tab in the New Physical Data Source dialog to enable import of a certificate key.
3. Click Import Certificate Key Store from File to import the certificate.

Studio displays a dialog to specify the certificate.

You can choose a jks or pkcs12 certificate key store for authentication between TDV and any Web data source that requires a trusted certificate.

4. Optionally, select a keystore from the list and click Clear Certificate Key Store to remove it.
5. Optionally, click Export Certificate Key Store to File to export the current certificate keystore to a jks or pkcs12 file.
6. Optionally, on the Advanced tab, set the Channel Pass-through field to a name or names that correspond to values passed in the HTTP request header for login authentication or for other purposes.

   The Channel Pass-through identifies the name of the HTTP request header property that is to be passed through to the WSDL, XML, or HTTP data source. You can specify multiple values, separated by commas.

   If the data source expects a property with a name different from what was originally sent in the HTTP request header, you can change the property name if you want. The name expected by the data source is put on the left side of the "=" operand, and the original property name is put on the right.
7. Optionally, on the Advanced tab, set the Environment Pass-through field to one or more environment variables to pass through to the WSDL, XML, or HTTP data source for login authentication or other purposes.

   Procedures can set an environment variable name and value by calling the SetEnvironment procedure. See the Info tab for /lib/util/SetEnvironment in the Studio resource tree for more information.

   Property names in the Environment Pass-through field can be renamed before they are passed to the data source, just as they can with channel pass-through.
8. Optionally, for REST data sources, specify the Execution Timeout (msec) period.

   Execution Timeout is the number of milliseconds an execution query on the data source is allowed to run before it is canceled. A value of zero milliseconds (default) disables the execution timeout. This can be used, for example, for resource-intensive cache updates set to run at non-peak processing hours.

TDV SOAP and REST OAUTH Examples

The use cases focus on how you would use TDV to customize sign-in automation, and configuration of the parts that do not conform to RFC 6749. These are examples—not exact, and not likely to run outside of one or two specialized environments. OAuth service providers occasionally change their sign-in process, which would require that you analyze the new sign-in process and design accordingly.

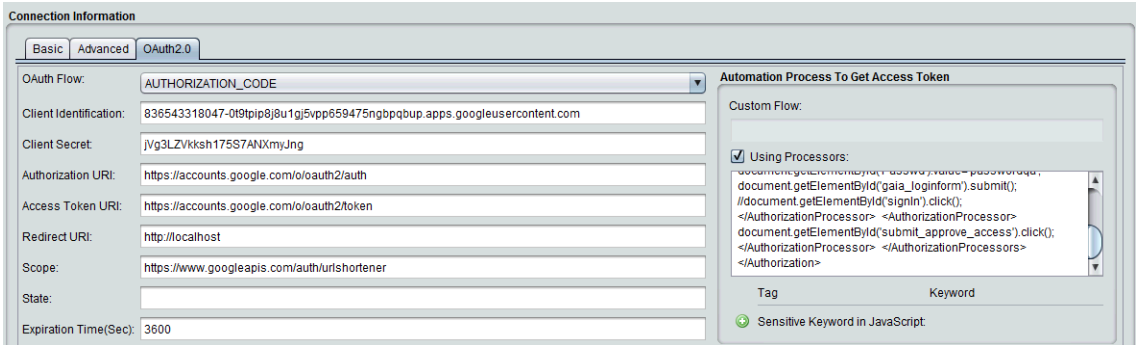
Google OAuth Example

With Google, every request and response follows RFC 6749, so the only thing to add is authorization.

If you do not want to show the password in clear text, create a tag like Testpassword instead of what is shown in the XML code for the editable text field below the Using Processors check box:

```
document.getElementById('Passwd').value=' Testpassword'';
document.getElementById('gaia_loginform').submit()
```

The sensitive tag is now Testpassword, and the sensitive keyword is the real password (xxxxxx).



OAUTH Tab Field	Example Value
OAuth Flow	AUTHORIZATION_CODE
Authorization URI	https://accounts.google.com/o/oauth2/auth
AccessToken URI	https://accounts.google.com/o/oauth2/token

Text field below the Using Processors check box	<Authorization>
	<AuthorizationProcessors>
	<AuthorizationProcessor>
	<![CDATA[
	document.getElementById('Email').value='test@gmail.com';
	document.getElementById('gaia_loginform').submit();
	//document.getElementById('next').click();]]>
	</AuthorizationProcessor>
	<AuthorizationProcessor>
	<![CDATA[
	document.getElementById('Passwd').value='xxxxx';
	document.getElementById('gaia_loginform').submit();
	//document.getElementById('signIn').click();]]>
	</AuthorizationProcessor>
	<AuthorizationProcessor>
	<![CDATA[
	document.getElementById('submit_approve_access').click();]]>
	</AuthorizationProcessor>
	</AuthorizationProcessors>
	</Authorization>

Facebook OAuth Example

Facebook does not conform to RFC 6749. According to RFC 6749, the request for the access token should be in FORM format, and the response should be in JSON format. With Facebook, the request is in QUERY format, and the response is in FORM format, so you need to use processors to configure them correctly.

In this example, Facebook's ExpireTime is named 'expires' instead of 'expires_in' as called for in RFC 6749, so the user should directly specify an expiry time.

Another way to get expire time is to use TokenProcessor, which can handle the input data and return standard JSON data. In this case, MessageValue is the value to retrieve from the response body, because the valid response is in FORM format. By retrieving access token and expire time from MessageValue, the token processor can return standard parameters that conform to RFC 6749 and JSON format.

OAuth Tab Field	Sample Values
Authorization URI	https://graph.facebook.com/oauth/authorize
AccessToken URI	https://graph.facebook.com/oauth/access_token
Text field below the Using Processors check box	<Authorization> <AuthorizationProcessors> <AuthorizationProcessor> document.getElementById('email').value='test@gmail.com'; document.getElementById('pass').value='xxxxxx'; document.getElementById('loginbutton').click(); </AuthorizationProcessor> </AuthorizationProcessors> </Authorization> <AccessToken> <RequestMsgStyle>QUERY</RequestMsgStyle> <ResponseMsgStyle>FORM</ResponseMsgStyle> <ExpireTime>1000</ExpireTime> </AccessToken> <TokenProcessor> VAR accesstoken; VAR expires; ...//Get access token and expire-time value from MessageValue MessageValue = "{access_token:" + accesstoken +", expires_in:" + expires+ "}"; </TokenProcessor>

Linkedin OAUTH Example

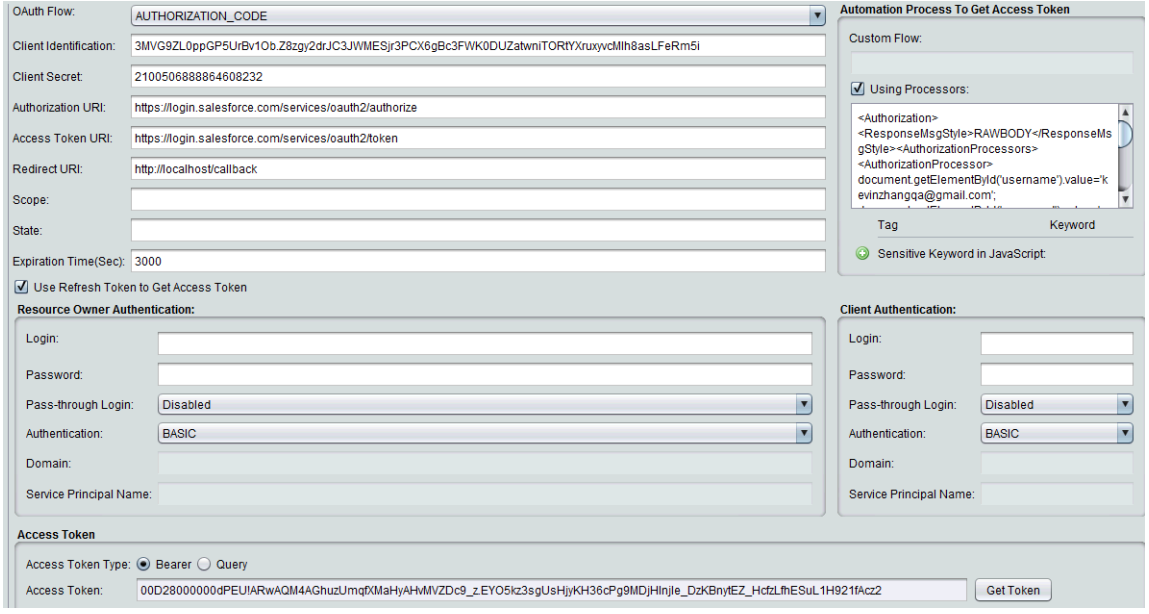
OAuth Tab Field	Sample Values
Authorization URI	https://www.linkedin.com/uas/oauth2/authorization
AccessToken URI	https://www.linkedin.com/uas/oauth2/accessToken

OAuth Tab Field	Sample Values
Text field below the Using Processors check box	<Authorization> <AuthorizationProcessors> <AuthorizationProcessor> document.getElementById('session_key-oauth2SAuthorizeForm').value='test@gmail.com'; document.getElementById('session_password-oauth2SAuthorizeForm').value='xxxxxx'; document.getElementsByTagName('form')[0].submit(); </AuthorizationProcessor> </AuthorizationProcessors> </Authorization>

Salesforce OAuth Example

Salesforce has some special requirements:

- When logging for an authorization code, Salesforce always sends email to a registered email box with a verification code, and asks for this code during the process. You need to handle this manually or use CustomFlow.
- Check Use Refresh Token To Get Access Token after obtaining the access_token and refresh_token the first time. If you do this, the token can be retrieved automatically each time, without needing to check the verification code manually in the email box.



In the example, using the JavaScript regular expression to fetch the authorization code is just for demonstration purposes.

OAuth Tab Field	Example Values
Authorization URI	https://login.salesforce.com/services/oauth2/authorize
AccessToken URI	https://login.salesforce.com/services/oauth2/token
Text field below the Using Processors check box	<Authorization> <ResponseMsgStyle>RAWBODY</ResponseMsgStyle> <AuthorizationProcessors> <AuthorizationProcessor> document.getElementById('username').value='test@gmail.com'; document.getElementById('password').value='xxxxxx'; document.getElementById('Login').click(); </AuthorizationProcessor> </AuthorizationProcessors> <TokenProcessor> var pattern=/\?code=.+'/i; var ans = pattern.exec(MessageValue); if(ans.index>0){MessageValue = "{code:" + ans[0].substring(6,ans[0].length-1)+"}";} </TokenProcessor> </Authorization>

Github OAuth Example

Github does not conform to RFC 6749 because the request is in QUERY format and the response is in FORM format.

OAuth Tab Field	Sample Values
Authorization URI	https://github.com/login/oauth/authorize
AccessToken URI	https://github.com/login/oauth/access_token

OAuth Tab Field	Sample Values
Text field below the Using Processors check box	<Authorization> <AuthorizationProcessors> <AuthorizationProcessor> document.getElementById('login_field').value='test@gmail.com'; document.getElementById('password').value='xxxxxx'; document.getElementsByTagName('form')[1].submit(); </AuthorizationProcessor> </AuthorizationProcessors> </Authorization> <AccessToken> <RequestMsgStyle>QUERY</RequestMsgStyle> <ResponseMsgStyle>FORM</ResponseMsgStyle> </AccessToken>

Foursquare OAuth Example

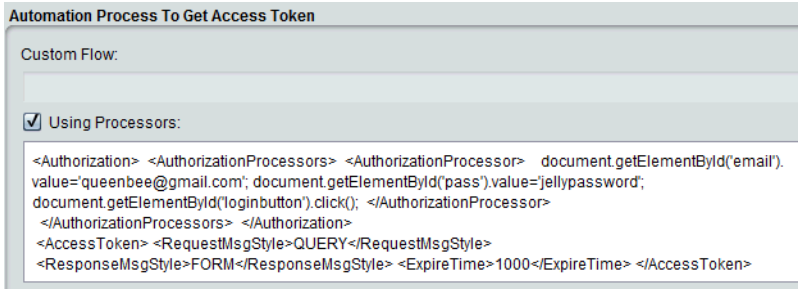
Foursquare uses Query as the access token type. However, it uses oauth_token as the access token name instead of using access_token in the URL, as defined in RFC 6749, so you need to use QueryTokenName for the service.

OAuth Tab Field	Sample Values
Authorization URI	https://foursquare.com/oauth2/authenticate
AccessToken URI	https://foursquare.com/oauth2/access_token
Text field below the Using Processors check box	<Authorization> <AuthorizationProcessors> <AuthorizationProcessor> document.getElementById('username').value='test@gmail.com'; document.getElementById('password').value='xxxxxx'; document.getElementById('loginToFoursquare').submit(); </AuthorizationProcessor> </AuthorizationProcessors> </Authorization> <QueryTokenName>oauth_token</QueryTokenName>

TDV OAuth Tab XML Processors Field Reference

On the OAuth tab for your WSDL, SOAP, and REST data sources, you can define custom processors. The custom processors use XML to provide authorization and access token customization. TDV provides a collection of processors and XML elements that you can use to accommodate nonconforming requests and responses. This topic provides examples that use these processors to illustrate how they obtain access tokens from several well-known third parties.

This topic is a reference of the XML element syntax that is valid for entry in this field.



The OAuth authorization framework enables a third-party application to obtain secure, temporary access to an HTTP service. However, because most interactions between client and resource owner do not conform to RFC 6749, it may be necessary for you to modify requests and responses to make them conform to the specification.

For example, you might have code similar to the following in the field:

```
<Authorization>
   <AuthorizationProcessors>
      <AuthorizationProcessor>
         document.getElementById('email').value='test@gmail.com';
         document.getElementById('pass').value='xxxxxx';
         document.getElementById('loginbutton').click();
      </AuthorizationProcessor>
   </AuthorizationProcessors>
</Authorization>

<AccessToken>
   <RequestMsgStyle>QUERY</RequestMsgStyle>
   <ResponseMsgStyle>FORM</ResponseMsgStyle>
   <ExpireTime>1000</ExpireTime>
</AccessToken>

<TokenProcessor>
   VAR accesstoken;
   VAR expires;
...//Get access token and expire-time value from MessageValue
```

```
   MessageValue = "{access_token:" + accesstoken +", expires_in:" +
expires+ "}";
```

</TokenProcessor>

- Authorization Element Reference, page 154
- AccessToken XML Element Reference, page 156
- RefreshToken Element, page 159
- QueryTokenName Element, page 161

Authorization Element Reference

The authorization element lets you customize the authorization segment of the OAuth flow.

XML Schema of the Element

```
<xs:element name="Authorization" minOccurs="0">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="RequestMsgStyle" minOccurs="0"/>
       <xs:element ref="ResponseMsgStyle minOccurs="0""/>
       <xs:element ref="AuthorizationProcessors minOccurs="0""/>
       <xs:element ref="TokenProcessor" minOccurs="0"/>
     </xs:sequence>
   </xs:complexType>
```

</xs:element>

Sequence Element	Description of value

RequestMsgStyle	According to RFC 6749, the default value is GET for Authorization Code Grant, Implicit Grant, and Customized Flow.
	`<xs:element name="RequestMsgStyle" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:enumeration value="FORM"/>`
	`<xs:enumeration value="QUERY"/>`
	`<xs:enumeration value="QUERYPOST"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`
	• QUERY—Append all request parameters of OAuth to the URL as a query string, using the HTTP GET method.
	• QUERYPOST—Add all request parameters of OAuth to URL as query string, using the HTTP POST method.
	• FORM—Add all request parameters of OAuth to the request body, using a Content-Type of application/x-www-form-urlencoded.

ResponseMsgStyle	According to RFC 6749, the default value is GET for Authorization Code Grant, Implicit Grant, and Customized Flow.
	`<xs:element name="ResponseMsgStyle" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:enumeration value="FORM"/>`
	`<xs:enumeration value="QUERY"/>`
	`<xs:enumeration value="JSON"/>`
	`<xs:enumeration value="RAWBODY"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`
	• QUERY—All response parameters of OAuth are returned as a query string appended to a redirect URL.
	• FORM—All response parameters of OAuth are returned with the entity, and Content-Type is application/x-www-form-urlencoded.
	• JSON—All response parameters of OAuth are returned with the entity, and Content-Type is application/json.
	• RAWBODY—All response parameters of OAuth are returned with the entity, but the format is not clearly defined. In this case, use tokenProcessor(JavaScript) to retrieve all parameters.

Sequence Element	Description of value
AuthorizationProcessors	Simulates a browser (user agent), automatically performing whatever authorization process is required to log in. Each AuthorizationProcessor within AuthorizationProcessors should be mapped to one pop-up page of the browser. <xs:element name="AuthorizationProcessors" minOccurs="0"> <xs:complexType> <xs:sequence> <xs:element maxOccurs="unbounded" ref="AuthorizationProcessor"/> </xs:sequence> </xs:complexType> </xs:element> <xs:element name="AuthorizationProcessor"> <xs:simpleType> <xs:restriction base="xs:string"/> </xs:simpleType> </xs:element>
TokenProcessor	Get tokens or other parameters if the response is not in the specified format. **XML Schema of the Element** <xs:element name="TokenProcessor" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:string"/> </xs:simpleType> </xs:element>

AccessToken XML Element Reference

The AccessToken element lets you customize the acquisition of an access token in the OAuth flow. A way to get expire time is to use TokenProcessor, which can handle the input data and return standard JSON data. A MessageValue can be used to retrieve from the response body, because the valid response is in FORM format. By retrieving access token and expire time from MessageValue, the token processor can return standard parameters that conforms to RFC 6749 and JSON format.

XML Schema of the Element

```
<xs:element name="AccessToken" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="RequestMsgStyle" minOccurs="0"/>
      <xs:element ref="ResponseMsgStyle" minOccurs="0"/>
      <xs:element ref="ExpireTime" minOccurs="0"/>
      <xs:element ref="TokenProcessor" minOccurs="0"/>
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
```

Sequence Element	Description of Values

RequestMsgStyle	According to RFC 6749, the default value is FORM for Authorization Code Grant, Client Credentials Grant, Resource Owner Password Credentials Grant, and Customized Flow.
	`<xs:element name="RequestMsgStyle" minOccurs="0">`
	`  <xs:simpleType>`
	`  <xs:restriction base="xs:string">`
	`  <xs:enumeration value="FORM"/>`
	`  <xs:enumeration value="QUERY"/>`
	`  <xs:enumeration value="QUERYPOST"/>`
	`  </xs:restriction>`
	`  </xs:simpleType>`
	`  </xs:element>`
	• QUERY—Append all request parameters of OAuth to the URL as a query string, using the HTTP GET method.
	• QUERYPOST—Add all request parameters of OAuth to URL as query string, using the HTTP POST method.
	• FORM—Add all request parameters of OAuth to the request body, using a Content-Type of application/x-www-form-urlencoded.

Sequence Element	Description of Values

ResponseMsgStyle	According to RFC 6749, the default value is JSON for Authorization Code Grant, Client Credentials Grant, Resource Owner Password Credentials Grant, and Customized Flow.
	`<xs:element name="ResponseMsgStyle" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:enumeration value="FORM"/>`
	`<xs:enumeration value="QUERY"/>`
	`<xs:enumeration value="JSON"/>`
	`<xs:enumeration value="RAWBODY"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`
	• QUERY—All response parameters of OAuth are returned as a query string appended to a redirect URL.
	• FORM—All response parameters of OAuth are returned with the entity, and Content-Type is application/x-www-form-urlencoded.
	• JSON—All response parameters of OAuth are returned with the entity, and Content-Type is application/json.
	• RAWBODY—All response parameters of OAuth are returned with the entity, but the format is not clearly defined. In this case, use tokenProcessor(JavaScript) to retrieve all parameters.

ExpireTime	Sets an expiration time for the access token, overriding the default time of 5 seconds.
	XML Schema of the Element
	`<xs:element name="ExpireTime" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:integer"/>`
	`</xs:simpleType>`
	</xs:element>

Sequence Element	Description of Values
TokenProcessor	Gets tokens or other parameters if the authorization response is not in the specified format. **XML Schema of the Element** <xs:element name="TokenProcessor" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:string"/> </xs:simpleType> </xs:element>

RefreshToken Element

The RefreshToken element lets you customize the way the access token is refreshed in the OAuth flow.

XML Schema of the Element

```
<xs:element name="RefreshToken" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="RequestMsgStyle" minOccurs="0"/>
      <xs:element ref="ResponseMsgStyle" minOccurs="0"/>
      <xs:element ref="ExpireTime" minOccurs="0"/>
      <xs:element ref="TokenProcessor" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

```
</xs:element>
```

Sequence Element	Description of Values

	According to RFC 6749, the default value is FORM for Authorization Code Grant, Resource Owner Password Credentials Grant, and Customized Flow.
	`<xs:element name="RequestMsgStyle" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:enumeration value="FORM"/>`
	`<xs:enumeration value="QUERY"/>`
	`<xs:enumeration value="QUERYPOST"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`
	• QUERY—Append all request parameters of OAuth to the URL as a query string, using the HTTP GET method.
	• QUERYPOST—Add all request parameters of OAuth to URL as query string, using the HTTP POST method.
	• FORM—Add all request parameters of OAuth to the request body, using a Content-Type of application/x-www-form-urlencoded.

	According to RFC 6749, the default value is JSON for Authorization Code Grant, Resource Owner Password Credentials Grant, and Customized Flow.
	`<xs:element name="ResponseMsgStyle" minOccurs="0">`
	`<xs:simpleType>`
	`<xs:restriction base="xs:string">`
	`<xs:enumeration value="FORM"/>`
	`<xs:enumeration value="QUERY"/>`
	`<xs:enumeration value="JSON"/>`
	`<xs:enumeration value="RAWBODY"/>`
	`</xs:restriction>`
	`</xs:simpleType>`
	`</xs:element>`
	• QUERY—All response parameters of OAuth are returned as a query string appended to a redirect URL.
	• FORM—All response parameters of OAuth are returned with the entity, and Content-Type is application/x-www-form-urlencoded.
	• JSON—All response parameters of OAuth are returned with the entity, and Content-Type is application/json.
	• RAWBODY—All response parameters of OAuth are returned with the entity, but the format is not clearly defined. In this case, use tokenProcessor(JavaScript) to retrieve all parameters.

Sequence Element	Description of Values
	Sets an expiration time for the access token, overriding the default time of 5 seconds. **XML Schema of the Element** <xs:element name="ExpireTime" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:integer"/> </xs:simpleType> </xs:element>
	Gets tokens or other parameters if the authorization response is not in the specified format. **XML Schema of the Element** <xs:element name="TokenProcessor" minOccurs="0"> <xs:simpleType> <xs:restriction base="xs:string"/> </xs:simpleType> </xs:element>

QueryTokenName Element

If the access token type is Query, this element specifies the name in the query string if the name is different from access_token.

XML Schema of the Element

```
<xs:element name="QueryTokenName" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:element>
```

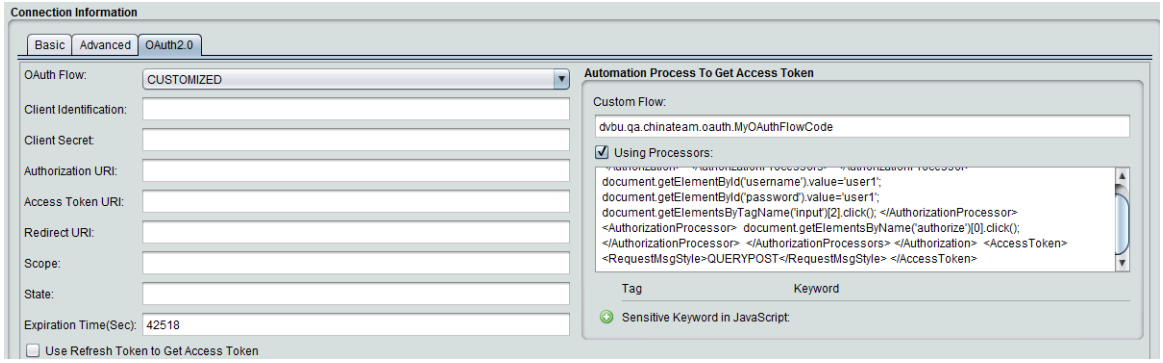
Partial Example of Using OAuth Customized Flow for WSDL, SOAP, and REST

The CustomFlow interface is made available to support customized options or processes so that they can obtain an access token and other parameters.

In OAuthProfileCallback, all required fields are provided in the UI. In each request:

- Construct the OAuthProfileCallback.Request with method info, request URL, request body and headers.
- Handle OAuthProfileCallback.Response to get results.

You might need to call handleAuthResponse several times while interacting with the service side. Because this is a simulation of the browser, it is your responsibility to extract the required information from OAuthProfileCallback.Response.




```
package com.compositesw.extension.security.oauth;
import java.util.Map;
/**Custom flow used for Extension Grants of OAuth2.0:
http://tools.ietf.org/html/rfc6749#section-4.5.
 * Any OAuth 2.0 flow with customized request or response that does
not conform to RFC 6749 can be
 * customized.
 * With CustomFlow, a flow step is ignored if the request is NULL,
or if no request info is defined
 * in OAuthProfileCallback.
 */
public interface CustomFlow {
    /**
    * Build authorization request. If request info is NULL, the
authorization step is ignored. */
    public void buildAuthRequest(OAuthProfileCallback callback)
throws Exception;
    /**
    * Handle authorization response. */
    public void handleAuthResponse(OAuthProfileCallback callback)
throws Exception;
    /**
    * Build access token request. If request info is NULL, the
authorization step is ignored.
     * The flow fails if both authorization request and access
token request info are NULL.
```

```
     */
    public void buildAccessTokenRequest(OAuthProfileCallback
callback) throws Exception;
    /**
     * Handle access token response. */
    public void handleAccessTokenResponse(OAuthProfileCallback
callback) throws Exception;
    /**
     * Build refresh token request. If request info is NULL, the
authorization step is ignored.
     */
    public void buildRereshTokenRequest(OAuthProfileCallback
callback) throws Exception;
    /**
     * Handle refresh token response. */
    public void handleRefreshTokenResponse(OAuthProfileCallback
callback) throws Exception;
    /**
     * All OAuth elements (access_token, refresh_token, expires_in,
token_type, scope, etc.)
     * extracted from response can be found in the value map
returned by getOAuthElements(). */
    public Map<String, Object> getOAuthElements();
    /**
     * Get access token. */
    public String getAccessToken();
    /**
     * Get refresh token. */
    public String getRefreshToken();
}
```

Configuring File Data Sources

This topic describes the configuration of file-based data sources. For the purposes of TDV, the file data sources grouped in this topic are those that are file based and require similar configuration options.

Supported Character Encoding Types

There are over 110 supported character-encoding types, they include:

• Cp1250	• utf-16be
• Cp1257	• utf-16le
• iso-8859-1	• windows-1250
• us-ascii	• windows-1251
• utf-8	• windows-1256
• utf-16	• windows-1257

Supported URL Protocols

Data sources that reference a file, can do so through a URL. TDV supports several URL protocols, including file, http:, https:, ldap, smb, and ftp. For example, your file data source could be located at one of the following URL locations:

- file:///C:/projectstuff/build/trialrun/teststuff/flatfile/USPSaddresses.csv
- file:///\\megawat/engineering/bees/swarmhive/xml_files/royaljelly_xml_1000.xml
- http://rss.news.queenbee.com/rss/topstories
- https://dvirtual-weakhive1/beepackage1/shadyhive10.csv
- ftp://ftp.varoa.fi/pests/standards/RFC/treatment_options.txt
- ldap://dvirtual-waxmoth:389/dc=small,dc=net
- http://dvirtual-waggle/cgi-bin/dance_GetVoters.cgi
- jdbc:PostgreSQL://queenhost:3406/cs030101?continueBatchOnError=false&useUnicode=true
- smb://server/share/frame/file

Type	Limitation
FTP	HTTP, HTTPS and FTP are supported for reading the data. File must be in text format and unzipped.
network location	The URL to the single file must be relative to the machine where TDV Server is running.
machine without a Web server	It must be mapped to the machine where TDV Server is running.

About Export and Import of Custom Java Procedures

Custom Java Procedure JARs are exported with a TDV full server backup (and when using the backup_export), although the tool backup_export -excludejars option can be used to omit those files when required.

An export exception: If the Custom Java Procedure makes use of one or more classpaths referring to other JAR files, those files must be backed-up or migrated separately because they are not picked up and replicated during export.

When a data source is exported, the adapter from which the data source was created is exported as well. In particular, all the files in the data source directory are included in the CAR file.

Custom Java Procedures (CJP) are normally imported into the directory conf/customjars/ when restoring TDV from the full server backup CAR.

The CJP cluster propagation is immediately propagated across the cluster along with its CJP library. Unlike other TDV-defined resources, those resources that can be referred to by a CJP data source's classpath are *not* propagated. Such resources must be manually distributed to all cluster nodes.

Adding a Custom Java Procedure

TDV has a JDBC interface and provides a bridge interface so that you can connect to a data source that is not currently supported. You can create a driver adapter that connects to that interface.

TDV supports custom procedures written in Java created to interface with other data sources. TDV provides APIs to create custom procedures.

A CJP library is a JAR file containing the Java classes implementing a set of Custom Java Procedures (CJPs) and other resources used by the CJPs. A CJP data source is a TDV custom data source that is created in Studio by specifying the signature of the CJP, a CJP library, and, optionally, a classpath. The classpath is needed only if the CJPs need resources that were not included in the CJP library.

For more details on TDV APIs to create custom Java procedures, see "JAVA APIs for Custom Procedures" in the *TDV Reference Guide.*

One adapter is sufficient to connect to any number of the same type of data sources. After it has been uploaded, the JDBC adapter functions like any other JDBC adapter, such as those used by Oracle, SQL Server, or MySQL. Customizations can be made to further change the adapter behavior.

You add a custom Java procedure to TDV Server as you would add a new data source. You must supply the specific JDBC driver and direct the server to the custom procedure JAR location so that TDV can upload it. The TDV server assumes that the JDBC adapter is implemented correctly. The server does not make any accommodations for JDBC adapters that do not supply correct metadata about the data source and it does not retrieve result sets that are not consistent with the metadata.

Note: If you need to export or import previously created custom Java procedures, see About Export and Import of Custom Java Procedures, page 166.

To add a custom Java procedure

1. Make sure that you have the Java code for the procedure.
2. Compile the Java code and put the compiled code into a JAR file.

   Your classpath should point at <TDV_install_dir>\apps\extension\lib\csext-xxx.jar, where 'xxx' is your most recent patch level.

   For instance, if TDV server is installed under C:\Apps\cis6.1 and you are running 6.1.0.00.24, then you would use:

   javac -classpath C:\Apps\cis6.1\apps\extension\lib\csext-1024.jar TestCJP.java
3. Add the class to a JAR file. For example:

   jar -cvf TestCJP.jar TestCJP.class
4. Place the JAR file on the machine where TDV Server is running.
5. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
6. In the New Physical Data Source dialog, select Custom Java Procedure as the Data Source Adapter and click Next.
7. Supply the information for a custom Java procedure data source:

   — Name—Name for the data source.

   — Custom procedure jar location—Use the Browse button to locate the path to the JAR file containing the procedures on the server, or type the full path to the JAR file.

   For example: file:///C:\myExamples\myProcedures.jar.

   The JAR can be uploaded *only* from a file location that is visible to the TDV Server.

   — Additional classpath—Optionally, specify any classpath that might be needed by the Java custom procedure class. To specify multiple classpaths, separate them with semicolons.

   For example, if the Java custom procedure uses classes contained in Widget.jar, you can type the path to Widget.jar, as follows:

   C:/composite/Widget.jar

8. Click one of these buttons:
   - Create & Introspect—To proceed immediately with introspection.
   - Create & Close—To create the data source; you can introspect at a later time.
9. See Retrieving Data Source Metadata, page 189 for how to introspect now or later.

Adding a Delimited File Data Source

A file-delimited data source is a file or set of files with value separators.

If the file does not have a header row, the column names are determined automatically.

If this data source is exported from a staging machine and imported to a production machine, the path for the logs directory might change from C:\<staging>\logs to C:\<production>\logs. Then, only the path to the root directory in the file data source needs to be modified after the data source is imported, and none of the queries to this data source need to be modified. After the root path is modified, it is your responsibility to re-introspect your data to ensure the existence of all the files. If the file structure of the new location is different from the old one, it entails adding/deleting/changing some of the files.

To add a file-delimited data source

1. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
2. In the New Physical Data Source dialog, select File-Delimited.
3. Click Next.
4. Type a name for the data source.
5. Select one of the following:
   - Local File System
   - URL
6. If the file is on the local file system, Select Browse and navigate to the root directory of the files for this data source.

   With this option, you can select one, more, or all the files in a directory. You can also select all the directories and all the files of the same type in those directories. However, even if all the files in the directory are of the

comma-separated values (CSV) type, detailed characteristics such as whether a header row exists must match.

7. If the files are located at a URL, specify the URL.

If file is at	Description
FTP URL	HTTP, HTTPS and FTP are supported for reading the data. The file must be in text format and unzipped.
network location	The URL to a single file must be relative to the machine where TDV Server is running.
machine without a Web server	It must be mapped to the machine where TDV Server is running.

8. Select Use Credentials if you want to specify user credentials here (rather than with system configuration) for connecting the data source.

Field	Description
Domain	User's domain; for example, composite.
User Name	Name of the user.
Password	User's password.

9. Accept the default or specify the Character Set encoding type.
10. Accept the default or specify the delimiter from among the following supported options:

    , (comma)
    : (colon)
    ; (semicolon)
    . (period)
    / (forward slash)
    \ (backward slash)
    <TAB> (horizontal tab—ASCII code character 09 hexadecimal)
    <SOH> (start of heading—ASCII code character 01 hexadecimal)

11. Accept the default or specify a Text Qualifier for which the whole text in the file is enclosed.
12. Accept the default or specify the number of the row in the file where the data begins.
13. Select the Has Header Row check box if the file text has a header row.

14. Select Ignore trailing delimiter check box if each row can contain a trailing delimiter that you want to ignore.
15. Accept the default file extensions to filter the root directory for, or type in the file extension values for which you want to filter. Example of two filters: `*.csv,*.txt` (a space is allowed between two filters)

    Rules for the filters:

    — `*` (asterisk) means that any character in the filename occurs zero or more times.

    — `?` (question mark) means that any character in the filename occurs exactly once.

    — `,` (comma) is a separator between filters.

    — `\` (backslash) is an escape character to escape a filename that contains * (asterisk), ? (question mark), or , (comma).
16. Click one of these buttons:

    — Create & Introspect—To proceed immediately with introspection.

    — Create & Close—To create the data source; you can introspect at a later time.
17. See Retrieving Data Source Metadata, page 189 for how to introspect now or later.

Adding an XML File Data Source

If this data source is exported from a staging machine and imported to a production machine, the path for the logs directory might change from C:\<staging>\logs to C:\<production>\logs. Then, only the path to the root directory in the file data source needs to be modified after the data source is imported, and none of the queries to this data source need to be modified. After the root path is modified, it is your responsibility to re-introspect your data to ensure the existence of all the files. If the file structure of the new location is different from the old one, it entails adding/deleting/changing some of the files.

After you have added a file-XML data source to the resource tree, you cannot change its file path. If you want a new file path, delete it and create it again.

To add a file-XML data source

1. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
2. In the New Physical Data Source dialog, select File-XML.
3. Click Next.
4. Type a name for the data source.
5. Select one of the following:
   - Local File System
   - URL
6. If the file is on the local file system, Select Browse and navigate to the root directory of the files for this data source.

   With this option, you can select one, more, or all the files in a directory. You can also select all the directories and all the files of the same type in those directories. However, even if all the files in the directory are of the comma-separated values (CSV) type, detailed characteristics such as whether a header row exists must match.
7. If the files are located at a URL, specify

If file is at	Important Information
FTP URL	HTTP, HTTPS and FTP are supported for reading the data. File must be in text format and unzipped.
network location	The URL to the single file must be relative to the machine where TDV Server is running.
machine without a Web server	It must be mapped to the machine where TDV Server is running.

8. Select Use Credentials if you want to specify user credentials here (rather than with system configuration) for connecting the data source.

Field	Description
Domain	User's domain; for example, composite.
User Name	Name of the user.
Password	User's password.

9. Accept the default or specify the Character Set encoding type. The Character Set drop-down list includes <auto-detect> as the default option for file-XML data sources, letting Studio detect the character set automatically.

10. Optionally, type in the location of the XML schema file using this syntax:
`<namespace> <location> [<namespace> <location>]`

    — <namespace> is the target namespace for the XML schema

    — <location> is the absolute path (including the name of the file) to the XSD file.

    — A white space is needed between the target namespace and location.

    If you want to use an external XSD file for resolving the schema, specify the location of the XSD file in the Schema Location field. If you want to let the system decide the XML schema for you, leave the Schema Location field blank.

    Example:
`http://www.compositesw.com/services/webservices/system/admin/resource file:///C:/test.xsd`

namespace	`http://www.compositesw.com/services/webservices/system/admin/resource`
location	`file:///C:/test.xsd`

11. Optionally, type in the No Namespace Schema Location to specify the location for an XML Schema that does not have a target namespace.

12. Accept the default file extensions to filter the root directory for, or type in the file extension values for which you want to filter.

    Rules for the filters:

    — `*` (asterisk) means that any character in the filename occurs zero or more times.

    — `?` (question mark) means that any character in the filename occurs exactly once.

    — `,` (comma) is a separator between filters.

    — `\` (backslash) is an escape character to escape a filename that contains * (asterisk), ? (question mark), or , (comma).

13. Click one of these buttons:
    - Create & Introspect—To proceed immediately with introspection.
    - Create & Close—To create the data source; you can introspect at a later time.
14. See Retrieving Data Source Metadata, page 189 for how to introspect now or later.

Adding a Cache File Data Source

The file-cache data source is used for storing resource data that are cached using the Automatic storage mode. For additional information, see TDV Caching, page 465. The file-cache data source uses a directory for each table in it, and a file in that directory for storing the data. The data files are binary encoded.

To add a file-cache data source

1. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
2. In the New Physical Data Source dialog, select File-Cache as the Data Source Adapter and click Next.
3. Type a name for the data source.
4. Click Browse and use the Path Selection dialog box to locate the path to the storage directory for the file cache data source.
5. Click one of these buttons:
    - Create & Introspect—To proceed immediately with introspection.
    - Create & Close—To create the data source; you can introspect at a later time.
6. See Retrieving Data Source Metadata, page 189 for how to introspect now or later.

Adding an LDAP Data Source

You can add an LDAP data source and configure it to behave like a relational table. During introspection, TDV maps all LDAP data types to the string data type.

The Active Directory objectGUID attribute displays in the "binding GUID string" format. For example, c208521a-6fcd-43f2-90ad-ed790c9715c1. If a value for the objectGUID comes from anywhere other than LDAP or is specified in a TDV view or script, that value must use the same dashed string format.

To add an LDAP data source

1. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
2. In the New Physical Data Source dialog, select LDAP and click Next.
3. Type a name for the data source.

   When the process of adding the data source is complete, this name is displayed in the Studio resource tree representing the data source.
4. On the Basic tab, provide this information:

- URL—Type the path to the LDAP data source in the URL field, in the following format:

  ldap://<host_name>:<port_number>/o=<organization_name>

  For example:

  ldap://platinum:370/o=earth.com

  The directory suffix depends on how the LDAP is set up: o for organization, ou for organizational unit, cn for common name, dn for distinguished name, or dc for domain component.
- Login—Valid username, if required, to access the underlying data source. When the data source is used as a target for cache tables or for data ship, the sign-in user must be granted the ability to create tables, execute DDL, and perform other tasks. In some cases, the LDAP connection does not require a username.

  Example of a username: cn=Ldap Manager
- Password—Valid password, if required, to access the underlying data source. In some cases, the LDAP connection does not require a password.
- Save Password—Check box is enabled only if Pass-through Login (further down in this window) is enabled. See Basic Settings for a Relational Data Source, page 87 for information about setting pass-through login, and About Pass-Through Login, page 77 for additional details.

- Authentication—Choose the method the LDAP client is to use to authenticate itself to the data source.
  - Simple: The client sends the LDAP server its fully qualified domain name and a clear-text password. This authentication mechanism can be used within an encrypted channel such as SSL, if it is supported by the LDAP server.
  - Digest
  - Kerberos
- Pass-through Login—Choose whether pass-through login is to be Enabled or Disabled. See Basic Settings for a Relational Data Source, page 87 for information about setting pass-through login, and About Pass-Through Login, page 77 for additional details.

5. Click the Advanced tab.
6. On the Advanced tab, provide this information:
   - Delimiter—Select a field delimiter from among the following supported options:

     , (comma)
     . (period)
     : (colon)
     ; (semicolon)
     / (forward slash)
     \ (backward slash)
     | (vertical bar)
   - Connection Pool Min Size—Minimum number of connections per connection identity (data source) that can be maintained concurrently (default 10).
   - Connection Pool Max Size—Maximum number of connections per connection identity (data source) that can be maintained concurrently (default 100).
   - Connection Pool Timeout (s)—Number of seconds (default 30) that a connection can remain idle in the pool without being closed and removed from the pool.
   - Execution Timeout (s)—Number of seconds an execution query on the data source is allowed to run before it is canceled. The default value of zero seconds lets even long processes run to completion.

7. Click one of these buttons:
   - — Create & Introspect—To proceed immediately with introspection.
   - — Create & Close—To create the data source; you can introspect at a later time.
8. See Retrieving Data Source Metadata, page 189 for how to introspect (now or later).

Adding Microsoft Excel Data Sources

There are two ways to introspect and use Microsoft Excel files. The method used depends on whether the TDV Server you are working with is hosted on a Windows operating system or UNIX operating system:

- Adding Microsoft Excel Data Sources, page 177
- Adding Microsoft Excel (non-ODBC) Data Sources, page 181

In both cases the Microsoft Excel files must be locally accessible to the TDV Server on a mapped or mounted drive. Flat files do not expose a JDBC interface, so direct (mapped or mounted) LAN access to those flat files is required.

If you want to introspect Excel documents that contain non-US characters, you should use the Non-ODBC Excel data source.

Note: Excel files are loaded into managed memory. If managed memory is insufficient, the introspection fails.

Adding Microsoft Excel Data Sources

The Microsoft Excel driver leverages Microsoft ODBC. Excel sheets and named areas within sheets are introspected as TDV tables.

For more information about semijoin fields, see the *TDV Administration Guide*.

To add a Microsoft Excel data source on Windows

1. Before introspecting the Excel data sheet as a TDV available data source, configure it as an ODBC-available data source with a DSN locally accessible to the TDV Server.
2. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
3. In the New Physical Data Source dialog, select Microsoft Excel and click Next.

4. Type a name for the data source.
5. On the Basic tab, provide this information for a Microsoft Excel data source:
   - DSN—Type the DSN (Data Source Name) that was defined when the data source was added to the host machine.
   - Character Set—Character encoding type. See Supported Character Encoding Types, page 165.
6. Click the Advanced tab.
7. On the Advanced tab, provide:—

Advanced Tab Options	Description
Connection URL Pattern	A URL pattern that functions as a template for generating an actual URL to connect to the physical data source. Modify this template per implementation requirements, but be aware that TDV does not validate modifications. The data source adapter may or may not validate changes, ignoring invalid URL connection parameters.
Connection URL String	The literal URL string that is generated from the connection URL pattern with the connection information you provide. This string is used by the JDBC adapter to connect to the physical data source. This field is generated by the system and is not editable. For further details, see "Configuring TDV Data Connections" in the *TDV Administration Guide*. When TDV is running on a Windows platform you can access a file located on the network with a file URL like the following: file://10.1.2.199/d$/public/folder/ExcelFileName.xls If you map a network drive from the computer hosting TDV to connect the computer to file resources, you can browse to the directory to introspect more than one Excel file at a time, or specify a file URL to add a single file. The following is an example of a Windows file URL: file:///Z:/shared/folder/ExcelFileName.xls
Connection Properties	Enables specification of property-value pairs that are passed to the targeted JDBC data source to determine the data source behavior for the connection with TDV. A selection of commonly used properties for all the supported versions of MySQL, Oracle, and Sybase are populated on the Advanced tab with default values.

Advanced Tab Options	Description
JDBC Connection Properties	Click to open a window in which to add custom JDBC connection properties for any JDBC data source. You can add multiple property-value pairs by clicking the Add Argument button, or delete pairs by clicking the Remove Argument button adjacent to them. TDV does not validate property names. The data source adapter might ignore incorrectly named properties or invalid values, or it might provide an error code.
Maximum Connection Lifetime (m)	Sets the duration, in minutes, that an inactive connection (a connection that was returned to the pool) persists if there are more open connections than the minimum pool size. The duration is calculated from the creation time not from the time that the connection was released to the pool. A value of "0" allows connections to last indefinitely. Default: 60 minutes.
Connection Validation Query	A native data source query that is sent directly to the data source without evaluation by the TDV query engine. Enter a query that gives a quick return. If the validation query returns a non-error result, this fact validates the connection to the data source.
Execution Timeout (s)	The number of seconds an execution query on the data source is allowed to run before it is canceled. The default value of zero seconds disables the execution timeout so processes that take a long time are allowed to run. For example, cache updates set to run at non-peak processing hours can be resource intensive processes that take much longer than a client initiated request.
Execute SELECTs Independently	If this option is checked, a SELECT statement submitted to this data source is executed using a new connection from the connection pool and committed immediately after the SELECT is completed. INSERT, UPDATE, and DELETE statements continue to be executed using the same connection as part of the transaction.

Advanced Tab Options	Description

Connection Check-out Procedure	Specify the procedure to return a valid SQL statement for that database which can be used to initialize the connection. One common case is to initialize Oracle Virtual Private Database (VPD)-based systems.
	
	VPD is a method of doing row-level security. Complex security policies can be set to allow or deny access to subsets of data in a table. After the connection is made, often with a generic account, the client enables certain sets of access rights by setting a security context. In this case, the init procedure returns something like dbms_session.set_identifier('username'). This would then be executed on the connection, changing the privileges associated with that connection from the default to those associated with the username passed.
	
	In addition, other parameters can be changed. A block like this might be returned by the init procedure:
	`BEGIN`
	`dbms_session.set_identifier('username');`
	`EXECUTE IMMEDIATE 'alter session set optimizer_index_cost_adj=10';`
	`EXECUTE IMMEDIATE 'alter session set optimizer_index_caching=90';`
	`EXECUTE IMMEDIATE 'alter session set "_complex_view_merging"=true';`
	`END;`
	
	This example code is Oracle-specific. Others databases have similar functions.
	
	The signature of the **`init`** procedure should look like this:
	`IN ds_name VARCHAR, OUT sqlText VARCHAR)`
	
	The code should be written such that the init procedure causes rights to be revoked if not called with the appropriate context.

Supports Star Schema	Check this option if this data source can support large predicates for star schema semijoins. Do not check this option unless you are sure the data source can support receiving queries with large predicates and large cardinalities. Refer to Star Schema Semijoin, page 587 for more information.
Max Source Side Cardinality for Semi Join	Sets the maximum source-to-source ratio of cardinality for semijoins.

Advanced Tab Options	Description
Min Target to Source Ratio for Semi Join	Sets the minimum target-to-source ratio of cardinality for semijoins.
Max Source Side of Semi Join to Use OR Syntax	Sets the maximum source-side use of the OR syntax for semijoins.

8. Click one of these buttons:
   - Create & Introspect—To proceed immediately with introspection.
   - Create & Close—To create the data source; you can introspect at a later time.
9. See Introspecting Data Sources, page 191 for how to introspect now or later.

Adding Microsoft Excel (non-ODBC) Data Sources

When adding Microsoft Excel files as data sources on UNIX platforms, use the Microsoft Excel (non-ODBC) data source adapter to begin configuration. The TDV Server uses a non-ODBC Microsoft Excel driver based on Apache POI to enable introspection and to use multiple Excel data files at one time.

To add a Microsoft Excel data source on a UNIX platform

1. Before introspecting the Excel data sheet as a TDV available data source, configure it as an ODBC-available data source with a DSN locally accessible to the TDV Server.
2. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
3. In the New Physical Data Source dialog, select Microsoft Excel (non-ODBC).
4. Click Next.
5. Type a name for the data source.

6. Select one of the following:

Selection	Description
Local File System	Select if the Excel file is on the local file system. With this option, you can select one, more, or all the files in a directory. You can also select all the directories and all the files of the same type in those directories to introspect all Excel spreadsheets at the same time. Specify the root path to begin the search for the files on the local file system. Root path is the absolute path to the root directory where the files reside.
URL	For TDV running on UNIX operating systems, you can add an Excel file located on the local machine with a URL file protocol like the following: `file:///usr/name/folder/excel_filename.xls` The directory containing the source Excel file must be mounted to the UNIX server hosting TDV. For example if the computer directory 10.1.2.199/d$/public contains the Excel file, it could be mounted as /root/public. The Excel file could be accessed with a file URL like: file:///root/public/folder/excel_filename.xls

7. Optionally, specify a filename filter to restrict what types of files are to be introspected. This adapter supports introspection and use of two kinds of Excel data source files: *.xls (Excel 97-2003) and *.xlsx (Excel 2007). During introspection, the introspection tree only displays the filenames with this extension. If you want to introspect more than one type of file, separate the type filters with commas.

   Rules for the filters:

   — `*` (asterisk) means that any character in the filename occurs zero or more times.

   — `?` (question mark) means that any character in the filename occurs exactly once.

   — `,` (comma) is a separator for each filter.

   — `\` (backslash) is an escape character to escape a filename that contains an asterisk, question mark, or comma.

   Note: Only files that match the filter you specify here are exposed later on during the process of adding or removing data source resources (through the Add/Remove Resources menu option) and also during data source re-introspection.

8. Optionally type values for or make selections for the following:

Element	Description
Character Set	Character encoding type. See Supported Character Encoding Types, page 165.
Data Range	Enter the value that indicates the data range you want to introspect.
Blank Column Type	Choose the data type to apply to blank columns: Varchar, Double, Boolean, or Datetime.
Has Header Row	Check if the first row of all the introspected Excel sheets has a row of column names. If it is not selected, the first row of each Excel data sheet is introspected as a data row and the column names are: COL1, COL2, COL3. After the connection is established and the Excel files are introspected, each sheet is made available as a TABLE that can be defined as having or not having a header row independently of the original schema header row setting.
Columns in Every Row Use Format Categories of Columns in First Row	Check to introspect the data in every row formatted as specified in the first row.
Ignore Invalid Data	Check to ignore invalid data.
Introspect with Formatted Display Values instead of Actual Values	Check to introspect the formatted display values.
Blank Value as Null Value	Check to introspect blank values as null values.

9. Click one of these buttons:
   - Create & Introspect—To proceed immediately with introspection.
   - Create & Close—To create the data source; you can introspect at a later time.
10. See Retrieving Data Source Metadata, page 189 for how to introspect now or later.

Configuring XML/HTTP Data Sources

An XML/HTTP data source collects data from raw XML over HTTP. It collects the information for a single XML/HTTP operation, writes the WSDL for you, and establishes the WSDL data source instance.

The following steps are involved in configuring an XML/HTTP data source for use:

- Creating an XML Definition Set, page 184
- Adding an XML/HTTP Data Source, page 185
- Retrieving XML Data Over HTTP, page 187

Creating an XML Definition Set

An XML definition set provides the output document definition for retrieving XML data over HTTP.

This section gives quick steps for creating an XML-type definition set. Quick steps for creating an XML definition set are given in this section. For more details, see Definition Sets, page 377.

To create an XML-type definition set

1. Right-click at an appropriate location in the resource tree, and select New Definition Set.
2. Type a name for the definition set.
3. Select XML in the Type drop-down list.
4. Click OK.

   The definition set is added to the resource tree, and the editor for the definition set opens in the right pane.
5. In the editor, select the XML Schema tab to define your XML schema. You can do this in one of three ways.
   - — You can type your XML schema in this panel.
   - — You can upload an existing XML schema by clicking the Import XML Schema Definitions From File(s) toolbar button, navigating to the file location, and selecting the XML schema file to import. To show or hide the

connection properties and alternate location mappings, check or uncheck the Show Advanced Options box.

— You can base your schema on an existing XML instance. Click the Create XML Schema Definitions from XML Instance toolbar button and navigate to the XML file. When you click Open, the XML file appears on the XML Schema panel of the editor.

Note: The XML instance should be an actual XML file, not a schema.

6. To format the definition set with hierarchical indentation, and with keywords in orange type and literal definitions in blue type, click the Format XML Schema Definitions toolbar button.
7. Click the Validate XML Schema Definitions toolbar button.

   If the schema is valid, an information box appears with the message, The definition set is valid. If not, an error dialog box opens with a message describing the line and column where a problem was encountered.

   Note: A field in the lower right corner of the XML Schema panel indicates the current line and column position of the cursor, separated by a colon; for example, 20:34 for line 20, column 34.
8. Save the new XML schema using Studio's File > Save command.

Adding an XML/HTTP Data Source

Use the steps in this section to add an XML/HTTP data source. After adding the data source, see Retrieving XML Data Over HTTP, page 187, for details on how to access the data.

To create an XML/HTTP data source

1. Right-click at a location in the Studio resource tree where you want this data source to reside, and select New Data Source.
2. In the New Physical Data Source dialog, select XML/HTTP and click Next.
3. Supply this connection information for an XML/HTTP data source:

   — Name—Name for the data source.

   On the Basic tab, provide this information for an XML/HTTP data source:

   — URL—URL to the HTTP server hosting the XML source.

   — Method—GET or POST

   Use GET to request data by sending the request information to the HTTP server. Typically, GET is used to retrieve a static resource. However, a query

string or extra path information can be appended as a text string after a question mark (?) in the URL of a GET request to trigger server-side processing.

Use POST to send data to the HTTP server to be processed, making sure that the data format of the sending and the receiving programs are compatible.

— Login—Valid username and password to access the HTTP server.

— Password—Password for the sign-in username to access the HTTP server.

— Save Password—Check box that is enabled only if Pass-through Login is enabled. See Basic Settings for a Relational Data Source, page 87 for information about setting pass-through login.

— Pass-through Login—See Basic Settings for a Relational Data Source, page 87, and About Pass-Through Login, page 77, for more information.

— Authentication—Choose the method of authentication for this data source.

— Domain—For NTLM authentication only, enter the domain.

— Service Principal Name—For NEGOTIATE authentication only, enter the service principal name.

— Access Token—For OAUTH, type the value of the token. For example:

```
e72e16c7e42f292c6912e7710c838347ae178b4a&scope=user%2Cgist&token_type=bearer
```

— Automatically Retrieve Access Token—For OAUTH.

— No Input—Select if no input is required.

— Input in URL—Select if the input is in the URL.

— Input Document Definition—To specify the input document definition, use the Browse button to locate the definition set you created as described in Creating an XML Definition Set, page 184.

— Output Document Definition—To specify the output document definition, browse to the path of the output document definition that supplies the schema for retrieving XML data, as in this example:

/<path to definition set>/xmlhttp_ds_schema/GetVotersResponse

When you click the Browse button for the Input Document Definition or the Output Document Definition, the Choose Schema window opens.

a. In the left pane, select the XML definition set in the resource tree.

b. In the right pane, make sure that Element is selected in the Show drop-down list, and select the definition.

When Element is selected, a specific schema response is expected by the XML/HTTP service. The fully qualified name of the document element must be the same as the element chosen from the schema.

When a Type is selected, the document element must be of the same type.

When you make your selection in the right pane, the Schema field is automatically filled with the complete path to the input or output document definition.

c. Click OK.

You return to the New Physical Data Source window.

4. Click the Advanced tab.
5. On the Advanced tab, enter this information:
   - Certificates—Enter information about the security certificates used for the XML/HTTP data source. You can click the buttons above the table to:
   - Import Certificate Key Store From File—Type or browse to the path, and select the type and password for the key store.
   - Export Certificate Key Store To File—Export a key store to a file.
   - Clear Certificate Key Store—Clear the key store.
   - Channel Pass-through—Identifies the name of the HTTP request header property that is to be passed through to the XML or HTTP data source. Multiple values can be passed through to the data source by specifying the names as a comma-separated list.
   - Environment Pass-through—Enter one or many environment variables to pass to the XML or HTTP data source for login authentication or other purposes.
6. Click one of these buttons:
   - Create & Introspect—To proceed immediately with introspection.
   - Create & Close—To create the data source; you can introspect at a later time.
7. See Retrieving Data Source Metadata, page 189, for how to introspect now or later.

Retrieving XML Data Over HTTP

After adding the XML/HTTP data source, if you want to have the XML data in tabular form to use it in SQL queries, you need to transform the XML data.

After creating an XML/HTTP data source, you can use it as follows to get XML data:

- Execute the XML source within the data source, and view the data in XML format.
- Create a transformation of the data source, and execute the transformation to view the XML data in tabular form.

To execute the XML source within the data source

1. Double-click the XML source (also called Web service operation) within the XML/HTTP data source.

Or, you can first select the XML/HTTP data source, and then select the File > Open <data source> menu option.

2. In the editor that opens on the right, click the Execute toolbar button.

   The result is displayed in the Result panel.

3. Click the Details button to view the XML data.

To create a transformation of an XML/HTTP data source

1. Right-click at an appropriate location in the resource tree, and select New Transformation.
2. In the Create Transformation window, select a non-XQuery transformation type. For example, select Basic XML to Tabular Mapping.
3. Click Next.
4. In the tree display, select the XML source in the XML/HTTP data source for the transformation.
5. Type a name for the transformation in the Transformation Name field.
6. Click Finish.
7. For further details on transformations, see Using Transformations, page 311.

Retrieving Data Source Metadata

To complete the data source creation process in Studio, the data source metadata must be introspected from the source. This topic describes how to introspect the data sources that you have configured, set filters for introspection, and reintrospect data sources.

The following topics are covered:

You can use the TDV API to set up introspection tasks and view the results. You can find the introspection API operations under:
`/Data Services/Web Services/system/admin/resource/operations/`

For instructions on how to use these operations, open the operation in Studio and click the Info tab, or see the *TDV Application Programming Interfaces Guide*.

About Introspection and Reintrospection

Introspection is the process of collecting native data source metadata so that all selected resources—like catalogs, schemas, tables, and procedures—are represented in a standardized way in the TDV virtual data modeling layer. The representations include capabilities and connection profiles. Composite views and procedures can then be built on the data modeling layer.

Introspection can be performed any time after data source configuration. When setting up introspection, you can also set filter criteria so that any new resources of interest are automatically introspected.

Introspected resources are added to the Studio resource tree for use in the Studio modeling layer, where you can then create views, procedures, and other resources based on those data resources. The metadata layer can be further enhanced by definition of indexes and primary keys, by discovery of data relationships, and by gathering of statistics about the data. The TDV query engine uses introspection data to generate efficient queries with source-aware algorithms.

See Retrieving Data Source Metadata, page 189 for more information.

After each data source instance is configured and introspected in Studio, you can use that instance with any other data source instance to create views and procedures that intermix them all.

Note: Privileges on objects in the Data Source are evaluated only at the time of introspection or later when a FETCH is called during statement execution. So if privileges on an object in the data source are lowered after introspection, the error will not be caught until actual statement execution.

Reintrospection is the process of updating the data modeling layer after a data source has changed. During reintrospection, new data resources can also be automatically introspected for the first time.

Reintrospection can be initiated in any of these ways:

- Immediately, at any time.
- Automatically, according to a schedule.
- Triggered by conditions that you specify.
- Invoked by a TDV API call.

You can set filters that automatically reintrospect based on criteria applied to a catalog, schema, node, or resource.

Introspection and reintrospection can be performed in the background. Studio provides status reports about the resources selected. Multiple introspection and reintrospection tasks can be initiated on many resources. After resources are introspected, reintrospection is needed only if resource metadata has changed.

The introspection steps are similar for all data sources. They fall into three phases, each of which affects performance:

- Understanding the Resource ID Fetching Phase, page 211
- Understanding the Introspection Plan Creation Phase, page 212
- Understanding the Introspection Plan Implementation Phase, page 212

Introspecting Data Sources

For information about configuring TDV-supported data sources see:

- Working with Data Sources, page 65
- Configuring Relational Data Sources, page 77
- Configuring Web-Based Data Sources, page 105
- Configuring File Data Sources, page 165

Introspection steps are described in the following sections:

- Invoking Data Source Introspection, page 191
- Introspecting a Data Source, page 192

Various methods are available for working with introspection, including:

- Tracking Introspection, page 200
- Tips on Introspecting Based on a CREATE TABLE Statement, page 210

Invoking Data Source Introspection

You can invoke data source introspection when you first create a connection to a data source or at any time after the data source has been created.

To invoke introspection when the data source is initially created

- Click the Create and Introspect button on the New Physical Data Source panel.

To invoke data source introspection at any time after the data source has

been created

Use any of these methods:

- Right-click the name of the data source in the Studio resource tree and select the Add/Remove Resources option.
- Select the data source in the Studio resource tree. Open the Resource menu and select Add/Remove Resources.
- Open the data source and click the Add/Remove Resources button.

If you request multiple introspection tasks for the same data source, the requests are queued and executed sequentially, with all but the current task listed as WAITING.

Introspecting a Data Source

The introspection steps include three phases:

- Fetching the resource information from the data source.
- Creating an introspection plan based on rules and filters.
- Introspecting the data source.

All of these phases are accomplished using the Data Source Introspection window.

To introspect a data source

1. Invoke the introspection process as described in Invoking Data Source Introspection, page 191.

   The Data Source Introspection window opens and loads a list of resources that it detects in the data source.

   The process of loading resources can take significant time, depending on the size of the data source, connectivity, and whether the resource information is cached within TDV. See Understanding the Resource ID Fetching Phase, page 211.

   While resource information is loading, you can start selecting resources for introspection by checking the boxes next to resource names. You can even start an introspection task prior to loading all of the resource information.

   Note: If you click a triangle to expand a container, TDV immediately makes an extra call to load the contents of that container. This allows you to prioritize the loading of resources.

2. Select the resources you want to introspect by checking the boxes next to them.

   If you select a parent folder, all of the resources it contains are selected.

   Note: If you select a catalog, schema, or parent node before the entire list of contained resources is loaded, only those child resources that were loaded at the time of selection are selected. In this case, you might want to recheck later that the entire list has been loaded.

   When you select a resource, a Properties tab on the right displays the boxes that can be checked to control introspection of new resources in the future. See Reintrospecting Data Sources, page 206.

3. Optionally, filter the display of detected resources as described below.

   A count of the number of detected resources that are visible in the resource list versus the total number of resources is shown at the top of the resource list (for example, 5 Visible of 6 Total).

   You can use the fields and buttons above the resource list to control what is displayed.

   When searching with the Find field, type part or all of the name of a resource if you want to view only resources starting with this string. The Find field search pattern has the following rules:

   — Search patterns are applied to the entire resource path and name.

   — Only matches are shown.

   — The characters in the search pattern do not need to be adjacent—only in the same order.

   — The search pattern is not case-sensitive.

   — You can use "/" (forward-slash) to control how containers match. Without a slash, the search pattern spans catalog, schema, and resource names. (For example: abc matches resource a/b/c.) With one or more slashes, matching characters must be found in the same positions relative to the slashes.

   — No wildcards are supported.

   When searching with the Show field, filter what is displayed in the resource list using one of the following options.

Option	What it shows
Introspectable	All resources that can be selected for introspection.

Option	What it shows
Introspected	The resources that have already been introspected.
Changes	Only the changes made since the dialog was opened, including new selections and clears.
Adds	Only resources that have been added since the dialog box was opened.
Removes	Only resources that have been selected for removal from TDV since the dialog was opened.

— When searching with the filter buttons, click any of the buttons to include or not include empty containers, check or clear all resources, or expand or collapse the list.

4. Optionally, click Refresh Resource List to refresh the list of resources that are available for introspection.

   Note: Any currently selected resources are unchecked during the refresh process.

   You must click Refresh Resource List if you have added new resources after caching metadata and you want to bring those new resources into TDV. Studio and the TDV Server cache the resource list, and this button forces both caches to be refreshed. By default, the list of resources is cached within Studio for the seven most recently introspected data sources during the current user session. This cache exists only in memory and is cleared when Studio is closed. Another cache exists within the TDV Server to improve resource load time. The cache within TDV Server persists across Server restarts.

5. Select how you want TDV to run introspection based on the resources it encounters using the check boxes at the bottom of the Data Source Introspection dialog.

Check Box	Description
Re-Introspect previously introspected resources	Check to introspect all data sources, including those already introspected. If this check box is checked and dimmed then you are working with a data source adapter that has not yet been upgraded to the new introspection framework. Select this check box if you want to update existing TDV metadata for resources that might have changed at the source. Clear this check box for faster response when adding a new set of resources. You can work with new resources more quickly if you do not perform unnecessary full reintrospections of all existing resources. This check box setting does not apply to reintrospection initiated by API, scheduled reintrospection, trigger-initiated reintrospection, or manually initiated reintrospection.
Allow partial introspection, omitting resources with errors	Check to allow the metadata of those resources that are introspected to be committed to the TDV repository even if other resources fail to introspect. If this check box is checked and dimmed then you are working with a data source adapter that has not yet been upgraded to the new introspection framework.
Stop introspection upon the first error	Check only if the data source adapter does not yet support the new introspection framework. If unchecked, you can view all errors and warnings in the Introspection Status report without prematurely ending the introspection of other resources. If this check box is checked and dimmed, you are working with a data source adapter that has not yet been upgraded to the new introspection framework.
Copy privileges from parent folder	Check if you want the data source resources to inherit the access privileges accorded to the parent resource.

6. Optionally, for SOAP data sources, expand and define properties for each of the nodes. Properties can include any of the following.

Property	Description
Detect New Resources During Re-Introspection	If you select this option, child resources added to the current resource subsequent to this introspection will be detected during reintrospection. If the current resource is a catalog and you add a schema to it subsequent to this introspection, the added schema is detected during reintrospection. For details on reintrospection, see About Introspection and Reintrospection, page 190. If you do not select this option, child resources added to the current resource subsequent to this introspection are *not* detected during reintrospection. If the current resource is a catalog and a schema is added to it subsequent to this introspection, that schema is *not* detected during reintrospection. For details on reintrospection, see Reintrospecting Data Sources, page 206.
Binding Profile Type	Allows specification of the HTTP transport protocol, literal encoding scheme, and document message style.
Use Endpoint URL	Check box that allows you to configure the endpoint URL. If it is not checked, the URL defined in the WSDL is used. The Endpoint URL displays the real URL of the SOAP data source, which might be different than the WSDL URL. If the specified URL is invalid, the introspection process fails for the SOAP data source. The WSDL data source fails until the request is sent.
Endpoint URL	The Endpoint URL displays the URL where clients can access the SOAP data source, which might be different than the WSDL URL. You can use this field to edit the endpoint URL.
Default Timeout (msec)	The number of milliseconds the connection waits before declaring a failure is the attempt is unsuccessful.
MTOM Enabled	Select to enable the use of Message Transmission Optimization Mechanism (MTOM), a method of sending binary data to and from Web services.

Property	Description
Fast Infoset Enabled	Select to allow conversion of XML files into XML Infoset. This option provides for more efficient serialization than the text-based XML format.
Timeout (msec)	The number of milliseconds the connection waits before declaring a failure is the attempt is unsuccessful.
Choose Input Envelope	Obsolete field for legacy WSDL data sources. If you do not to select this option, the Choose Top Level Child Elements appears.
Choose Top Level Child Elements	Select this option to map the operation's request and response to input and output parameters, respectively. This option is configurable for every message part separately.
JMS related	If the SOAP data source connects through JMS, your node might display several fields that allow you to enter JMS information such as: • JMS Connector • JMS Destination • JMS Delivery Mode • JMS Expiry • JMS Priority

7. Optionally, expand and define introspection properties, which can include any of the following.

Property	Description
Detect New Resources During Re-Introspection	Detect child resources added to the current resource after this introspection. If the current resource is a catalog and you add a schema to it after this introspection, the added schema is detected during reintrospection. See About Introspection and Reintrospection, page 190. If you do not select this option, child resources added to the current resource subsequent to this introspection are *not* detected during reintrospection. If the current resource is a catalog, and a schema is added to it after this introspection, that schema is *not* detected during reintrospection. See Reintrospecting Data Sources, page 206.

Property	Description
Wildcard Symbol for Single Character Match	Defaults to an underscore.
Wildcard Symbol for Zero or More Character Match	Defaults to a percent symbol (%).
Escape Character for Wildcard Symbols	Defaults to a backslash (\).
Filter in Case Sensitive Mode	Check box that indicates your preference for running filters.
Separator for Each Filter	Defaults to a comma (,).
New Resource <Schema | Catalog | Procedure | Table> Name Filter(s)	Allows you to save the filter settings with a name that you specify.
Character Set	Character encoding type.
Schema Location	Enter the schema location using this syntax: `<namespace> <location> [<namespace> <location>]` <namespace> is the target name space for the XML schema. <location> is the absolute path (including the name of the file) to the .XSD file.
No Namespace Schema Location	The URL (one only) of a schema document that does not define a target name space.
Delimiter	Defaults to comma (,).
Text Qualifier	Defaults to asterisk (*).
Starting Row	Default to 1.
Has Header Row	Check box is selected by default.

8. Optionally, you can set up filters for the resources, but they are applied only if reintrospection is done. See Setting Introspection Filters on a Resource, page 203, for more information.
9. Click Next.

Studio opens a Data Source introspection plan that lists the resources that are to be introspected, based on your selections. The Data Source Introspection plan lists all the introspection changes (additions, removals, and updates) that will occur when you click the Finish button.

— Resources in black will be introspected.

— Resources in green will be added. (Their parent resources are also shown in green.)

— Resources in gray will be removed.

10. Review the introspection summary, and if necessary, click Back to revise the introspection plan and how it is to be done.
11. Click Finish to introspect the resources.

    TDV introspects the selected resources and adds the metadata about those resources to the TDV repository.
12. You can click OK to dismiss this window at any time; however, the introspection process continues running in the background. The introspection process continues even if you close the Studio session.

Panel	Shows
Running Tasks/Completed Tasks (left side)	The progress of introspection tasks and their status in the current user session.
Introspection Summary (upper right)	The status; introspection start and end times; number of warnings and errors; and the number of resources that were added, removed, updated, and skipped (plus an overall total).
Details (lower right)	The specific resources that were added, updated, removed, and skipped in the selected introspection. Those resources with no changes in the metadata used by TDV are added to the Skipped count. You can: • Show options—Filter the details displayed to view all details, only warnings, or only errors. • Show Details—Select any added resource then click show details to get additional information about the resource. • Export—Click to save a text introspection report that lists the tables, columns, and indexes that were introspected. See Viewing the Introspection Report for a Data Source, page 201 for an example of this file.

13. You can view the introspection information for a data source at a later time. See Tracking and Viewing the Introspection Process, page 200.

Tracking and Viewing the Introspection Process

Tracking Introspection

You can track time usage by the introspection process with the support of log files. A configuration parameter controls debug logging.

To diagnose introspection

1. Select Administration > Configuration from the main Studio menu.
2. Locate the Debug Output Enabled for Data Sources configuration parameter.
3. Set the parameter value to True.
4. After introspection, open cs_server_events.log to see how much time the various introspection steps used.

Viewing the Introspection Task Status Report

When you introspect a data source, TDV saves the introspection status information for all introspections done in the current user session so that it can be viewed at any time. The Introspection Task Status report displays the status of the data source introspections as well as the summary and details for the latest introspection that was performed.

You can view the introspection summary and details for any previously introspected data source at any time. See Viewing the Introspection Report for a Data Source, page 201 for more information.

To view the Introspection Task Status report

1. Open the introspection report:
   - From the Studio Resource menu, select Introspection Task Status.
   - If any introspection task is still running, you can click the introspection task progress bar at the bottom margin of the Studio application window to open this window.
2. Review the details of the introspection task status.

Panel	Shows
left	The status of all completed introspection tasks during this user session. A status of SUCCESS indicates that at least one resource was introspected. It does not mean that no errors or warnings were logged, but that the attempt was successfully committed to the data source.
right	The status summary and details for the most recent introspection.

Viewing the Introspection Report for a Data Source

When you introspect a data source, TDV saves the introspection information for the data source so that it can be viewed at a later time. The Introspection Report displays the summary and details of the most recent successful introspection of this data source.

To view the introspection results for a data source

1. Open the data source in Studio.
2. Click the Introspection Report tab.

   The Introspection Report tab displays the results of the latest introspection. It alerts you to any warnings or errors that might need your attention. It also indicates how many resources were introspected to synchronize changes with the data source.
3. Use the Show drop-down list to choose whether to display all warnings or errors. This selection filters the results to show any resources that had a problem.

   Sometimes network latency or other data collisions can cause a problem with resource introspection; these problems can be cleared up by attempting to introspect those resources again. At other times, a data source type incompatibility might cause a table column to fail to load.

Refer to the *TDV Reference Guide* for details about the data source data types that are supported by TDV.

4. Optionally, select a message in the Details list and click Show Details.

   TDV shows details about that message.

5. Optionally, click Export to export text file that contains a list of introspected resources.

Introspecting Data Source Table and Column Comment Metadata

Data sources, such as Oracle and Teradata, include the ability to add annotations or comments for added understanding. During the introspection process, TDV can retrieve this information and add it to the annotations field for each resource. The annotation field can be viewed on the Info tab of each resource.

Requirements

Requires one of the following data sources:

- Oracle
- DB2 LUW
- Teradata
- Netezza
- PostgreSQL
- DB2 mainframe
- Vertica, Composite
- SAP HANA
- HSQLDB
- Hbase

You can enable or disable the comments introspection for Oracle data sources.

To enable comment introspection

For data sources other than Oracle, comment introspection is performed by default.

1. Open the Oracle data source for which you want table and column level metadata retrieved.
2. Select the Advanced tab.
3. Locate and select the Introspect comments check box.
4. Run the introspection process.
5. Review the comments by opening a resource editor and selecting the Info tab.

Setting Introspection Filters on a Resource

When you introspect or reintrospect a data source, you can set filters to control how TDV runs the reintrospection process. You can set filters at the data source, catalog, or schema level to:

- Filter by catalog, schema, table, or procedure name.
- Use wildcard characters to filter names.
- Detect new resources during introspection.
- Filter in case-sensitive mode.

The filters are applied during reintrospection, whether invoked by API or trigger, at a scheduled time, or manually. Filters do not apply on *initial* introspection but are applied when during reintrospection.

Note: These filters are only available on relational data sources.

To set introspection filters on a resource

1. Invoke Add/Remove Resources from the context menu for a data source.
2. In the Data Source Introspection window, select a resource in the list on the left.
3. Optionally, in the Properties tab on the right, check Detect New Resources During Re-Introspection.
   — If you select this option, child resources added to the current resource subsequent to this introspection will be detected during reintrospection.

   If the current resource is a catalog and you add a schema to it subsequent to this introspection, the added schema is detected during reintrospection. For

details on reintrospection, see About Introspection and Reintrospection, page 190.

— If you do not select this option, child resources added to the current resource subsequent to this introspection are *not* detected during reintrospection.

If the current resource is a catalog and a schema is added to it subsequent to this introspection, that schema is *not* detected during reintrospection. For details on reintrospection, see Reintrospecting Data Sources, page 206.

4. Optionally, choose the filter settings for each resource.

   The filters displayed in the Properties tab on the right depend on whether you have selected a data source, catalog, or schema. For example, if you selected a catalog in the resource list, you can supply the value for filtering the schemas in the catalog in the New Resource Schema Name Filter(s) field. To filter for all schemas that start with the string SYS, you would type SYS%.

   For a data source, Studio displays the filters shown in the figure below. The Properties tab for a selected data source displays wildcard symbols you can use to filter the names of catalogs, schemas, tables, procedures, and folders. For a schema, Studio displays a smaller collection of filters.

   The possible fields on the Properties tab are described below:

Field or button	Description
Wildcard Symbol for Single Character Match	Symbol that stands for a single character. The default symbol is the underscore (_) and this cannot be changed. For example, if you type ab_ in the Schema Name Filter(s) field, it matches schema names such as abc and abd, where c or d occurs in place of the underscore.
Wildcard Symbol for Zero or More Characters Match	Symbol that stands for zero or more characters. The default symbol is the percentage sign (%) and this cannot be changed. For example, if you type ab% in the Schema Name Filter(s) field, it would match schema names such as abc, abd, ab, and any name with ab as prefix.
Escape Character for Wildcard Symbols	Character for escaping the symbols specified in the previous two fields (Wildcard Symbol for Single Character Match and Wildcard Symbol for Zero or More Characters Match). The default escape character is backslash (\ ) and this cannot be changed. For example, the sequence ab_c would match the resource-name ab_c by escaping the underscore (_).

Field or button	Description
Filter in Case-Sensitive Mode	Check to make the filter pattern case-sensitive. This option is not available for certain data sources. In such cases, the underlying physical data source determines the case-sensitive mode, and filters the names of its resources as well for introspection.
Separator for Each Filter	Symbol for separating filters. The default separator is a comma (,). For example, if you type orders, customers, it would match the table names orders and customers.
New Resource <resource type> Name Filter(s)	Where <resource type> can be a catalog, schema, procedure, or table depending on what is selected in the resource list. For example, if you select a data source that has no catalogs or schemas but does have procedures and tables, then only New Resource Table Name Filter(s) and New Resource Procedure Name Filter(s) are displayed on the Properties tab. Enter filters on the Properties tab to introspect only the resources that meet the filter criteria. See the preceding descriptions for wildcard symbols, escape character for wildcard symbols, and separator for filters. If no filter is specified, all new catalogs/schemas/procedures/tables (and their contents) are added during reintrospection. Existing resources that were not selected during the initial introspection or during a manual introspection are ignored during reintrospection, even if they match the New Resource Catalog <resource type> Name Filter(s) string. You can add previously ignored resources at any time using Add/Remove Resources. For example, if BZHANG is not selected for introspection and a reintrospection filter of B% is set for the data source, subsequent reintrospection finds any new schemas that begin with B but does not introspect BZHANG. The filters do not use the same serial character matching pattern as the Find field, which applies to the Introspectable list only. Strings are matched using the case sensitivity setting and the wildcard symbols specified.

5. Click Next.

The Data Source Introspection summary page lists all the introspection changes (additions, removals, and updates) that will occur when you click the Finish button.

— Resources in black will be introspected.

— Resources in green will be added. (Their parent resources are also shown in green.)

— Resources in gray will be removed.

The summary also lists at the bottom the rules that are applied during introspection as a result of what you selected for the check boxes on the previous panel.

6. Click Finish to initiate the introspection.

Reintrospecting Data Sources

Reintrospection is the process of synchronizing the TDV Server metadata representation of an introspected data source with the current state of the data source.

You can invoke reintrospection at any time, schedule it to occur periodically, use triggers, or call for it using a system API. Automated reintrospections can either just report on metadata differences or automatically reintrospect to synchronize TDV metadata to the current state of the resource.

Depending on the data source type (relational or file), all changes are detected, including the following:

- Changes in column data types
- Removed tables or columns
- Added tables or columns

Persistent reintrospection filters can be set to detect and incorporate new resources that meet the filter criteria during reintrospection. The filters are defined during introspection by the properties in the right side panel of the Data Source Introspection window.

Resources that were explicitly introspected remain introspected. Resources that were not selected during a previous introspection are not introspected, even if they happen to meet the criteria of a filter intended to find and introspect new resources.

The following sections describe the ways you can invoke reintrospection:

- Reintrospecting a Data Source Immediately, page 207

- Scheduling Reintrospection, page 207
- Triggering Reintrospection, page 208

If you are introspecting large data set, see this section for additional information:

- Tips for Reintrospecting Large Data Sets, page 214

If you are reintrospecting an SAP BW data set, see the *TDV SAP BW Adapter Guide*

Reintrospecting a Data Source Immediately

You can reintrospect a data source immediately at any time.

To reintrospect a data source immediately

1. You can reintrospect immediately in two ways:
   - — Right-click a data source name in the resource tree, and select Re-Introspect Now.
   - — Open the data source in Studio. Use this method if you want to review the data source definition or the scheduled reintrospection settings.
2. At the bottom of the Studio workspace for the open data source, click the Re-Introspection tab.
3. In the Immediate Reintrospection section, click Re-introspect Now.

   The Enter Password window opens if the password to access the data source was not saved when the data source was originally added to the server. For further details, see the descriptions for the Save Password and Pass-through Login fields under Adding Relational Data Sources, page 85.
4. If necessary, supply the password that is required to access the data source, and click OK.

   TDV immediately reintrospects the data source and displays the Introspection Task Status window with the introspection status, summary, and detailed results. See Viewing the Introspection Task Status Report, page 200 for more information.

Scheduling Reintrospection

You can schedule reintrospection to occur at a specific time, date, or interval. The scheduled reintrospection option is available only if the password was saved when the data source was originally added to the server.

To schedule reintrospection

1. Right-click a data source name in the resource tree, and select Open.
2. Click the Re-Introspection tab at the bottom of the Studio workspace for the open data source.
3. In the Scheduled Re-Introspection section, select the options for the schedule you want reintrospection to follow.
   a. Select Save Detected Changes if you want the changes detected in the data source during reintrospection to persist (that is, be saved in the repository). Uncheck this box if you want to inspect the changes prior to accepting them and committing data source metadata changes to the TDV metadata.
   b. Select the frequency with which you want reintrospection to repeat:
   — None (the default). Reintrospection is not scheduled.
   — Repeat Every < > minute(s), and type the number of minutes in the text field.
   — Repeat Hourly for the reintrospection to recur every hour.
   — Repeat Daily for the reintrospection to recur every day.
   — Repeat Weekly for the reintrospection to recur every week.
   c. Specify the starting time and date for the introspection in the corresponding drop-down lists.

   The date entered indicates the time at which the first occurrence of the reintrospection is to occur. For example, if a daily event is set for 11:55 a.m. three days in the future, it will run at 11:55 a.m. in three days and then every day thereafter.

   d. To send a reintrospection report to one or more email addresses, type and confirm the email addresses of the recipients.

   Separating multiple email addresses with commas (,) or semicolons (;).
4. Right-click on the resource panel's upper tab and select Save to save your settings.

Triggering Reintrospection

You can set up a trigger to reintrospect based on a JMS event, a system event, timer event, or user-defined event. See Triggers, page 391 for more information about triggers.

To set up a reintrospection trigger

1. Create a new trigger:
   a. Right-click in the Studio resource tree where you want the trigger to reside.
   b. Type a name for the new trigger and click OK.
2. Specify the condition that is to invoke the trigger: JMS Event, System Event, Timer Event, or User Defined Event.
3. For Action Type, select Reintrospect Data Source.
4. Specify the data source in one of these ways:
   — Type the resource-tree path and data source name in the Data Source Path field.
   — Click Browse and in the Select a datasource dialog box navigate to the data source, highlight it, and click **OK**.
5. Optionally, specify email information (in the To, Cc, Bcc, Reply-To, Message Subject, and Message Body fields) to send email when the trigger fires.
6. Select one or both of these options:
   — Do Not Persist Detected Changes—If you do not want to persist detected changes.
   — Do Not Send If No Changes Detected—If you do not want email sent if no changes were detected.
7. Check Enable Trigger near the top of the panel to enable this trigger.
8. Use Studio Manager to track the status and history of when this trigger runs.

Tips on Introspecting Multiple Files in a Network Share Folder

If the root path does not show the network mapped drives, the Root Path can be specified as UNC path without using the Browse button.

1. Define a new or edit an existing Microsoft Excel (non-ODBC) data source.
2. On the Basic tab for the Local File System value, if the Browse button does not allow you to navigate to the machine or root directory that contains your Excel data you can type the root directory location in the Root Path field. For example:

```
R:/<mymachinename>/work/users/mega/excel-ds
```

Note: The direction of the slashes in the root path are important. If you know your files are in a particular directory and they do not show in the introspection window, modify the direction of the slashes in your root pathname and save the data source.

3. Save the Microsoft Excel (non-ODBC) data source.
4. For an existing data source, on the Basic tab, scroll to and click the Add/Remove Resources button.
5. Multiple files in the network share folder should now be available for introspection.

Tips on Introspecting Based on a CREATE TABLE Statement

Under some circumstances, you can significantly reduce introspection time by collecting metadata based on the issued CREATE TABLE DDL statement rather than by introspecting it from the database after the table has been created. A configuration parameter is available for doing this.

To introspect a data source based on CREATE TABLE statement

1. Select Administration > Configuration from the main Studio menu.
2. Navigate to Server > Configuration > Debugging > DDL > Introspect newly created table using data source introspection.
3. Set the parameter value to True.
4. If you use the DDL statement to introspect, you can only examine table columns, primary keys, and indexes. Also, the results may be different from data source introspection.

Tips on Introspecting Large Data Sources

Large data sources with complex schemas and big tables with lots of columns have a correspondingly large amount of metadata that can pose challenges for TDV. For TDV, it is not the amount of data stored, but rather the complexity and amount of metadata that defines the data source that is the most important factor for performance, particularly during introspection.

Introspection can be broken down into three phases.

- Understanding the Resource ID Fetching Phase, page 211

- Understanding the Introspection Plan Creation Phase, page 212
- Understanding the Introspection Plan Implementation Phase, page 212

Understanding these phases can improve introspection performance for large data sources:

- Tips to Improve Performance of Resource ID Fetching, page 213
- Tips to Improve Performance of Introspection Implementation, page 213
- Tips for Reintrospecting Large Data Sets, page 214

Each of these phases has independent performance considerations.

Understanding the Resource ID Fetching Phase

So that TDV has complete metadata about all of the resources in your data set, do at least one full data source scan to retrieve all possible object names (resource identifiers) from within the data source. The initial fetch loads the resource list with basic metadata for all catalogs, schemas, tables, and procedures.

This phase corresponds to Invoking Data Source Introspection, page 191. After you have invoked introspection, TDV begins populating the resource list in the Data Source Introspection dialog with the resources it detects.

When obtaining this resource ID list, keep these things in mind:

- The list of resource IDs is cached within the TDV so that future fetches will be faster unless the list is recreated.
- Retrieving this list is asynchronous so that after the resources you need are visible within Studio, you can choose those resources before the full scan finishes.
- See Tips to Improve Performance of Resource ID Fetching, page 213 when introspecting large data sources.

The actual fetching of Resource ID from data sources is multithreaded to provide concurrency. One thread per container (data source root, schema, catalog, or other type of container) is created. The degree of parallelism (maximum number of threads) is either the number of processors, or half of the data source maximum connection pool size, whichever is less. Degree cannot be less than 1.

After the resource IDs are fetched and placed within a cache in the TDV, the resource IDs are directly fetched from the cache, unless the cache has been invalidated and needs to be recreated. This cache-based fetch is single-threaded but fast. The results of the fetch are streamed to Studio.

The resource identifiers are invalidated and the resource ID list is recreated when:

- You choose Refresh Resource List in the Data Source Introspection dialog when you choose Add/Remove Resources.
- Introspection (see Understanding the Introspection Plan Implementation Phase, page 212) is executed with an introspection plan that requests detection of new resources to be added.
- You invoke the deprecated or UpdateDataSourceChildInfosWithFilter (or the deprecated UpdateDataSourceChildInfos) Web services operation.

Reintrospecting existing resources does not recreate the cache unless the user has requested that new resources be detected for adding.

Understanding the Introspection Plan Creation Phase

After you have fetched the complete list of resource IDs (see Understanding the Resource ID Fetching Phase, page 211), you are ready to select the resources to introspect; that is, create an introspection plan (Introspecting a Data Source, page 192).

Loading resource names for a large data source can take a long time, so TDV lets you choose resources to introspect and drill down into as soon as their names are visible.

For container resources selected in the Add/Remove Resource dialog box, check the Detect New Resources During Re-Introspection check box under Properties on the right.

Some resources (most commonly the container resources within relational data sources) provide filter property options: New Resource Table Name Filter(s) and New Resource Procedure Name Filter(s). These filters only limit resources newly found during reintrospection.

Resources can be added through recursion using the TDV API.

Understanding the Introspection Plan Implementation Phase

After you select the resources to include and specify any filters for newly found resources, TDV displays a list of the resources in the introspection plan.

The introspection plan is displayed. You can change it by clicking Back and changing the filters and other settings as described in Introspecting a Data Source, page 192.

After completing the list of resources to be introspected, click Finish and the introspection task is launched. The introspection task status is displayed, along with a summary of the actions taken.

If you close Studio before finishing, introspection continues to run in the background.

The following describes the introspection and performance considerations:

- Introspection happens concurrently, one thread per container.
- Batching is applied to further improve performance.
- For each container, TDV evaluates the contents and breaks it into chunks of resources that have a common parent container and resource type. These chunks are broken into pieces smaller than the maximum introspection batch size. If this size is not defined for the data source, the chunks are divided into several pieces equal to the number of processors, or half the data source maximum connection pool size, whichever is less. This parallelism degree can be overridden within the data source capabilities file using introspect.override_parallelism_degree.

  Note: This override is only available for introspection, not resource ID fetching.

Tips to Improve Performance of Resource ID Fetching

If you are introspecting a very large data source, you can improve the performance during resource ID fetching by:

- Putting the TDV Server on a machine with many CPUs.
- Increasing the Connection Pool Maximum Size setting in Studio. Open the data source, select the Configuration tab, and select the Advanced tab.

  Note: The physical data source might be impacted if this number is too high.
- Allocating additional memory for both TDV Server and Studio if the data source you are introspecting has large amounts of metadata. (Large amounts of data do not matter.) To do this, increase the memory settings in these Studio configuration parameters: Privilege Cache Size (On Server Restart) and Metadata Cache Size (On Server Restart).

  Note: Do not request that a second cache be built while the initial cache of resource IDs is being built. If you do this, TDV does not attempt to merge these requests; instead the work is duplicated in parallel.

Tips to Improve Performance of Introspection Implementation

To improve introspection performance, you can:

- Put the TDV server on a machine with many CPUs.

- Increase the Connection Pool Maximum Size setting of the data source (be aware that the physical data source can get impacted if this number is too high). To change this setting, open the data source, and modify settings on the Advanced tab.
- Set the introspect.override_parallelism_degree in the data source capabilities file. The data source capabilities files for all supported data sources can be found at:

```
<TDV_install_dir>\apps\dlm
```

  Setting introspect.override_parallelism_degree in the data source capabilities file to a number higher than the number of CPUs might degrade performance. However, if there is network latency between the TDV and data source, setting this to a number higher than the number of CPUs might keep the TDV busy doing what it can while it waits for responses from the data source. This number cannot be higher than the maximum pool size of the data source. For example, this is a good option if you do not want to change the current connection pool size, but want to consume three-quarters of the pool instead of half of it.
- Specify introspect.max_batch_size in the data source capabilities file to tune introspection batching to match the batching capabilities of the data source. (Batching is supported for DB2, Oracle, PostgresSQL, Teradata, and Oracle E-Business Suite data sources. The default max_batch_size is 999.)

Tips for Reintrospecting Large Data Sets

The process for reintrospecting data sources is described in Reintrospecting Data Sources, page 206. When reintrospecting large data sets:

- If TDV has introspected a large number of resources for a given data source, introspecting this source again is an expensive operation.
- Reintrospection brings all new resources that have been added to the source after the most recent introspection, if the Detect New Resources During Re-Introspection is selected on the Data Source Introspection/Properties tab.
- If you want to add only a few new resources to the TDV, it might be better to newly introspect the resources rather than reintrospect them, because introspection only targets the selected resources.

Views and Table Resources

This topic describes the Studio user interface that you use to create and work with composite views and table resources.

About Composite Views

A composite view is a virtual data table defined by SQL and TDV metadata. A view defines a SQL query that comprises a SELECT statement and any ANSI-standard SQL clauses and functions. A view can JOIN or UNION to any resource defined in the TDV virtual data layer.

A view is designated in the resource tree with the View icon.

For more information about the View UI, see View and Table Editor Panel Reference, page 661

Views can be selectively published to make them available through client applications or the Web. If you want to make a view available to client programs, you must publish that view. See Publishing Resources to a Database Service, page 416 and Publishing Resources to a Web Service, page 421 for more information.

For information on scheduling a view's execution, see Creating a Timer Event Trigger, page 399.

Creating a New View

This section describes how to create a view that uses relational data source tables.

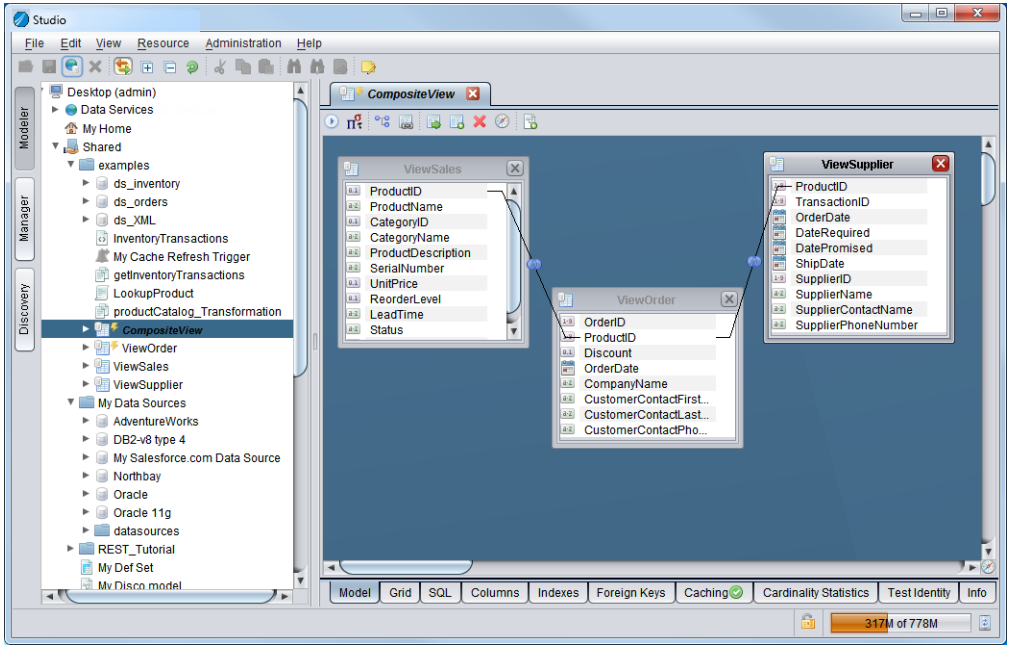
For details on selecting a location to create a resource, see Locating a Container for a TDV Resource, page 40.

To create a view

1. Right-click an appropriate location in the resource tree, and select New View, or use File > New > View.
2. In the Input dialog, type a name for the view.
3. Click OK.

   The view is added to the specified location in the resource tree, and the view editor opens the Model Panel in the right pane.

Studio provides a view editor that opens automatically when you create a view. You can open the view editor at any time by double-clicking a view's name in the resource tree.



4. Follow the procedure described in Designing a View and Table Resource, page 220.

Setting Default Query Options

Query hints are options that you can include to optimize your SQL statement for the TDV query engine. You can specify any of the options that are documented in "TDV Query Engine Options" in the *TDV Reference Guide*.

Options specified in a query override any defaults set using the configuration parameter values.

To specify global query hints

1. Select Administration > Configuration.
2. In the Configuration window, navigate to Default SQL Options.
3. Click the green plus button.

4. Type an option or hint type value for the Key. For example, INDEX.
5. Type a Key Value. For example, employees emp_dept_ix.
6. Click OK, to save the changes and exit the Configuration screen.

Commenting SQL

You can insert standard SQL comments by starting each line with a pair of hyphens. You can also insert multiple-line C-style comments, using /* to start the comment and */ to end the comment. Each of these methods is described below.

To insert standard SQL comments

1. Use one of these methods:
   - In the SQL panel, type two hyphens at the beginning of each line you want to be a comment line.
   - Select the lines you want to comment and type Ctrl-Hyphen, which adds two hyphens to the start of each line.

   You can type Ctrl-Hyphen again to remove the hyphens.
   - Select the lines you want to comment, and choose Edit > Comment Block.

   You can type Edit > Comment Block again to remove the hyphens.

   Note: If the selected block contains some lines that are already commented, another pair of hyphens is added at the beginning of the lines.

Although comments are stripped from the SQL request at runtime, a TDV configuration setting can cause comments to be passed through to the data sources so that client metadata can be recorded at the source. For details, see SQL Request Annotation Pass-Through, page 219.

To insert C-style comments

1. In the SQL statement, type `/*` to start the comment and `*/` to end the comment.

Adding Column Level Annotations

Annotations that are added to tables and views, are carried through when you publish the resource. After publishing, the column level annotations can be viewed in Business Directory or through the other client interfaces that you use to access published TDV data.

To set TDV to pass through SQL annotations

1. On the Studio resource tree, select the table or view for which you want to add column level annotations.
2. Right-click and select Open.
3. Type the text of your comment in the Annotations field.
4. Save your work.
5. Add annotations to other columns if you want to.

SQL Request Annotation Pass-Through

SQL annotations (comments) sent in requests to data sources can be used for customer implementations to enhance logging, traces, and tracking. By default, TDV trims SQL annotations from client requests.

To set TDV to pass through SQL annotations

1. On the Studio toolbar, select Administration > Configuration.
2. In the Configuration window, navigate to TDV Server > SQL Engine > SQL Language > Keep Comments.
3. Select the True radio button.
4. Click Apply.
5. Click OK to close the Configuration window.

Designing a View and Table Resource

View design involves performing one or more of these tasks, depending on your implementation requirements:

- Add resources for the view.
- Join tables by columns and specify join properties.
- Combine the output of multiple SQL selects.
- Specify global query options.
- Specify query options and query engine hints.
- Specify column output to include in or exclude from the view execution result.
- Apply functions on columns to transform result set values.
- Supply aliases for column names.
- Specify the sorting order for columns.
- Specify GROUP BY, HAVING, or EXPRESSION options.
- Specify constraints for the WHERE clause.
- Specifying indexes, primary keys, and foreign keys.

You can accomplish these tasks using the Model Panel, Grid Panel, SQL Panel, Columns Panel, Indexes Panel, and the Foreign Keys Panel.

The following sections describe how to perform these tasks:

- Designing a View in the Model Panel, page 220
- Designing a View in the Grid Panel, page 230
- Designing SQL for a View in the SQL Panel, page 237
- Generating a Model from the SQL Panel, page 241
- Defining Primary Key for a View or Table in the Indexes Panel, page 238
- Defining a Foreign Key for a View or Table Resource, page 239

Designing a View in the Model Panel

The Model panel is a graphical tool that helps you create SQL statements (without having to write the code) to obtain useful query results from disparate data sources.

To design a view

1. Select data resources for the view by dragging and dropping tables, procedures, and transformations into a view.
2. JOIN tables by linking table columns.
3. Configure JOIN properties
   a. Define the JOIN logical operators.
   b. Specify the preferred join algorithms or suggest semijoin optimizations.
   c. Provide estimates of table cardinality to help the query engine optimize the execution plan.
   d. Include all rows from the left, the right, or both sides of the join.
   e. Force join ordering or swap the order of the tables in the join.
4. Create UNIONs of SELECT statements, bringing together rows from matching tables.
5. Navigate among SELECT statements and the UNION of those SELECTs.
6. Use query hints to set the maximum number of rows to return, to set case sensitivity, to ignore trailing spaces, to mark the query as STRICT (adhere to SQL-92), or to force the query processing to disk if the query is likely to exceed memory constraints.

The following topics have more information on designing views in Studio:

- Adding a Resource to the Model Panel, page 221
- Joining Tables in the Model Panel, page 222
- Enforcing Join Ordering, page 224
- Creating a Union, page 225
- Navigating between Tables in the Model Panel, page 227
- Specifying the DISTINCT Query Option, page 228
- Specifying Query Hints, page 228

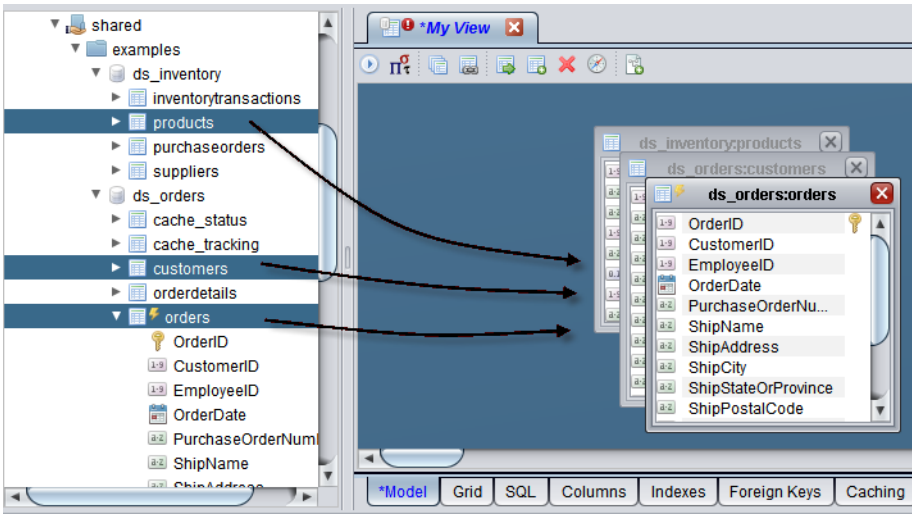
Adding a Resource to the Model Panel

You can add resources such as database tables, text files, procedures, and other views to the Model panel. Only a procedure that contains either all scalar outputs or just one cursor output can be included in a view.

To add a resource to the Model panel

1. Create a new view as described in Creating a New View, page 216.
2. Locate a resource in the resource tree.
3. Drag and drop the resource onto the Model panel.

   Note: You can use Ctrl-click or Shift-click to select multiple resources, and drag and drop them in the Model panel.



4. Rearrange the resources as needed.

Joining Tables in the Model Panel

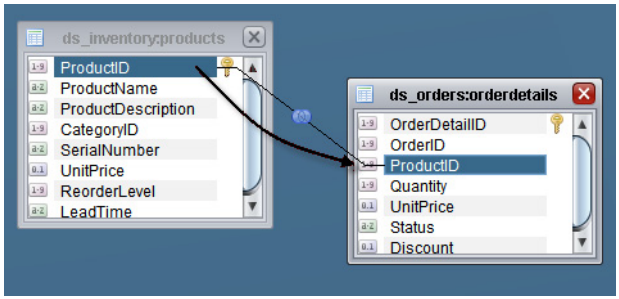
A view that you design might involve joining tables. You can often improve query performance by refining the properties of a join process including:

- Specify how the joined columns should be compared.
- Specify the join algorithm and cardinality.
- Choose a semijoin optimization.
- Specify whether to include all rows from a specific table.

In the Model panel, the join between the selected tables is visible as a line connecting those tables. This type of join is called an INNER JOIN. For details on tuning options, see Performance Tuning, page 551

To JOIN tables in the Model panel

1. Select the column from one of the tables to be joined. The first column selected is the "left" side of the join, unless Swap Order is selected in the Join Properties window.
2. Hold down the mouse button and drag from the selected column in the left-side table to the column to join it with in what will be the right-side table.



3. Make sure that columns specified for JOINS are of compatible data types.

   To check the data type of a column, click the Columns tab and open the Columns Panel.
4. Right-click the join icon on the JOIN line and select Properties to open the Join Properties window, or double-click the diamond graphic.
5. Select how the columns should be compared (=, <=, >, and so on) from the drop-down list in the top center of the Join Properties window.
6. In the Include rows section, check the boxes to specify the rows you want to include in the join:
   - Select the upper box to specify the `LEFT OUTER JOIN`.
   - Select the lower box to specify the `RIGHT OUTER JOIN`.
7. In the Join Details section:
   a. From the Specify Join Algorithm drop-down list, select the algorithm to use for the join.

   For the descriptions of the different algorithms, see Semijoin Optimization Option, page 579.

   b. Specify the Left Cardinality constraint.

   Provides cardinality hint for the left side of a join. It should be a positive numerical value, for example 50.

   c. Specify the Right Cardinality constraint.

Provides cardinality hint for the right side of a join. It should be a positive numerical value, for example 500.

d. If you select the Semijoin Optimization check box, the TDV query engine attempts to use the results from the number of rows to be processed for the join is minimized, and the query engine's performance is enhanced.

e. Select one of the following order options:

— Default Ordering—Applies the default ordering.

— Swap Order—Swaps the left and right sides of a join.

— Force Join Ordering—Overrides join ordering optimization.

f. Click OK to save the join specification.

8. Click the SQL tab to verify how the join is specified in the SQL statement, similar to the following example:

```
SELECT
    products.ProductID,
    orderdetails.UnitPrice,
    orderdetails.Status
FROM
/shared/examples/ds_inventory/products products FULL OUTER {
OPTION SEMIJOIN="True", LEFT_CARDINALITY="50",
RIGHT_CARDINALITY="500" } JOIN
        /shared/examples/ds_orders/orderdetails orderdetails
    ON products.ProductID = orderdetails.ProductID INNER JOIN
        /shared/examples/ds_orders/orders orders
    ON orderdetails.OrderID = orders.OrderID
```

9. Click OK.

Enforcing Join Ordering

You can allow the query engine to reorder the query execution plan based on statistics gathered from the data source table and optimization algorithms. However, given knowledge of the table contents, it is often advantageous to force the processing order derived from the written SQL.

Note: Joins that contain any user specified options cannot be reordered.

You can direct the query engine to follow the order in which you joined the tables. For more information, see SQL Join Reordering, page 576.

To enforce the processing order of table joins

1. Join the tables in the order you want.
2. Right-click the join line and select Properties. See Joining Tables in the Model Panel, page 222.

3. In the Join Properties window, select Force Join Ordering.
4. This step adds the phrase {OPTION FORCE_ORDER="true"} to the FROM clause and informs the query engine to query the tables in the order in which they are specified in the FROM clause.

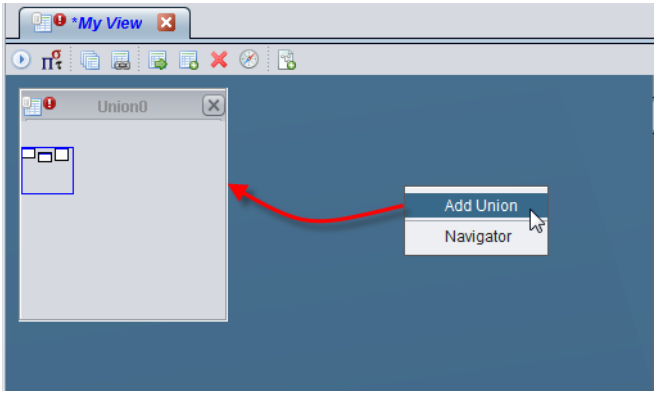
Creating a Union

Studio lets you graphically create a UNION between two or more tables.

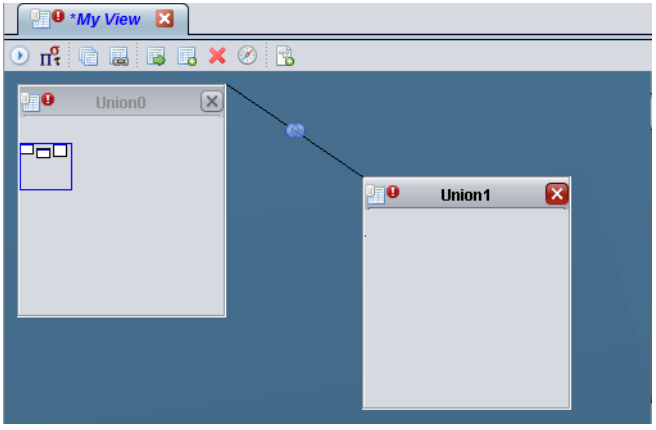
To graphically create a UNION

1. Create the first table for the UNION by dragging the resource onto the Model panel (if it is not already there).
2. Right-click in the background field of the Model panel and choose the Add Union option.

   The resources are condensed into a Navigator window called Union0. The tables are shown as smaller icons within the Navigator window.

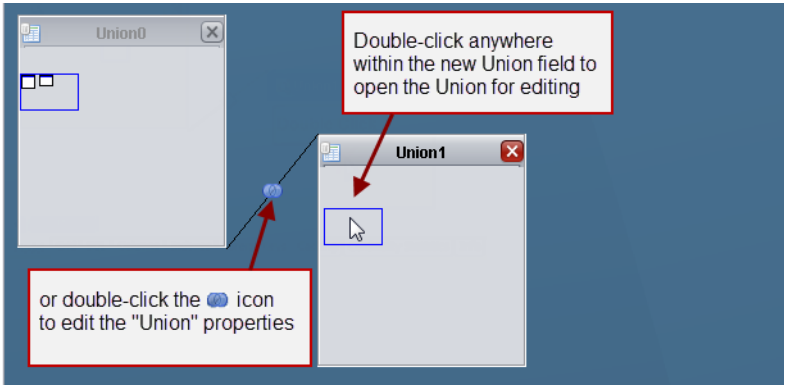


3. Right-click again in the Studio design space and choose the Add Union option to open a second UNION design space, called Union1.



4. Double-click anywhere in Union1 to open the UNION for editing.

   When you double-click Union1, the entire design space represents Union1, and is blank so that you can add resources to include in the UNION in the next step.



5. Drag and drop resources into the Union1 side of the SELECT statement.

   The definition of the two tables in the UNION must have the same number of columns in the same order, and they must be of compatible data types, for the UNION to function correctly.

6. After defining the second table for the UNION, right-click anywhere in the design space and select UP to return to the top-level query view navigator.

   By default the two tables are combined with a UNION ALL, but you can change that to a UNION, EXCEPT, or INTERSECT.

7. Double-click the UNION connector to open the UNION properties.
8. Make your design choices in the Union Properties window.
   a. By default, the two tables are combined with a UNION ALL (the All check box is checked by default), but you can change that to a UNION, EXCEPT, or INTERSECT.
   b. Optionally, check the PARALLEL, FORCE DISK, or DISABLE PUSH check boxes.

   For a description of TDV supported SQL (UNION, EXCEPT, INTERSECT, the All option) and the available query engine UNION options, see the *TDV Reference Guide*.

   c. Click **OK** to save the union properties.
9. Save the view.

Controlling the Number of UNION ALL and JOIN Flips

To optimize its execution plan, the query engine can flip between UNION ALL and various kinds of JOINs: INNER JOINs, LEFT OUTER JOINs, and RIGHT OUTER JOINs. If the two SELECT statements do not involve duplicates, the distributive law works for UNION DISTINCT and INNER JOINs as well.

By default, the maximum number of flips between UNIONs and JOINs is 2. You can change this maximum, although increasing it can have a negative impact on memory consumption and plan generation time.

To change the maximum number of flips between UNION ALL and JOIN

1. From Administration in the Studio main menu, select Configuration.
2. Navigate to Server > SQL Engine > Overrides.
3. Change the Value from its default of 2 to another integer.
4. Click OK at the bottom of the panel.

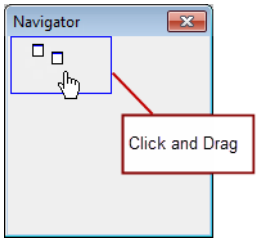
Navigating between Tables in the Model Panel

The Model panel is a large area that can hold many tables. Only a portion of the Model panel is visible at any given time. You can adjust the visible area using the scroll bars or by the Navigator window.

To navigate among the tables in the Model panel

1. Open the Navigator window by clicking the Navigator button.

   The Navigator window opens, displaying a miniature version of the Model panel with all its tables in position.



   After you first click in the Navigator window, a blue rectangle appears, showing what area is currently visible in the Model panel.

2. Click inside the blue rectangle and drag it over the area you want to see.

Specifying the DISTINCT Query Option

You can easily add the DISTINCT option to the SQL query in the Model panel. This query option ensures that duplicate rows are ignored.

To specify the DISTINCT query option in the Model panel

1. Right-click anywhere in the Model panel.
2. Choose Select Distinct.

   This selection adds **DISTINCT** to the view's **SELECT** statement, as in the following example:

```
SELECT DISTINCT
*
FROM
/shared/examples/ds_inventory/products products
```

3. Select the SQL panel and verify that the SQL statement now contains the DISTINCT option.

Specifying Query Hints

Query hints are options that you can include to optimize your SQL statement for the TDV query engine. Options specified in a query override any defaults set using the configuration parameter values.

You can specify the following query hints for the current view in the Model panel:

- Set the maximum number of rows to be fetched
- Set case sensitivity
- Set whether or not to ignore trailing spaces
- Force the Query Engine to use disk instead of memory for temporary query data storage
- Require that mathematical and string functions adhere to SQL-92 (STRICT)

Note: Many other query options can be added directly in the SQL panel. These are documented in "TDV Query Engine Options" in the *TDV Reference Guide*.

To specify query hints in the Model panel

1. Add however many tables you want in the Model panel.
2. Right-click anywhere in the Model panel, and select Query Hints.
3. In the Query Hints dialog box, specify the maximum number of rows to be fetched in the Max Rows Limit field.
4. Select one of the following options in the Case Sensitive drop-down list:
   - Server Default—Default setting of the server.
   - True—This option sets comparisons to case-sensitive mode.
   - False—This option sets comparisons to ignore case.
5. Select one of the following options in the Ignore Trailing Spaces drop-down list:
   - Server Default—Default setting of the server.
   - True—With this option, comparisons ignore trailing spaces.
   - False—With this option, comparisons do not ignore trailing spaces.
6. Select any check boxes that apply:
   - Force Disk—If selected, this option forces the query engine to use disk instead of memory wherever possible.
   - Strict—If selected, query engine does not push mathematical and string functions to the data source if the data source does not follow SQL-92 standards for those functions. This might affect performance.
7. Use the Clear All button if you want to clear all entries.
8. Click OK after setting the options.
9. Click the SQL tab to view the resulting options included in the SQL, as in the following example:

```
SELECT { OPTION MAX_ROWS_LIMIT="50",
IGNORE_TRAILING_SPACES="True", CASE_SENSITIVE="True",
STRICT="true", FORCE_DISK="true" }
    DISTINCT products.ProductID,
    products.ProductName,
    products.UnitPrice
FROM /shared/examples/ds_inventory/products products
```

Designing a View in the Grid Panel

The following sections describe Grid panel tasks. Drawing from the tables you added using the Model panel, use the **Grid** panel to select which columns to add:

- Listing All Columns from a Table, page 230
- Adding an Individual Column, page 231

After the columns have been added, you can use the Grid panel to refine column definitions, sorting order, and other attributes:

- Creating an Alias for a Column, page 232
- Changing a Column Data Type using the CAST Function, page 232
- Including a Column in the ORDER BY Clause, page 232
- Specifying a GROUP BY Clause, page 233
- Specifying Criteria on a Column, page 233
- Including a Function in a SELECT or WHERE Clause, page 234
- Declaring a Variable in a SELECT Statement, page 234

Listing All Columns from a Table

You can list all columns from a table in the Grid panel. These tables are added to the **SQL SELECT** statement. You can refine their definitions using the Columns panel.

To add all columns from a table as rows in the Grid panel

1. Click the List Columns button.

   The List Columns from Tables dialog box opens, displaying a field on the left showing the available tables, and a field on the right showing the tables you have selected.

2. You can move tables to the right side in one of two ways:
   - — Click the double-headed arrow button to move all available tables to the right side.
   - — Select one (or more, using Ctrl-click or Shift-click) of the available tables, and click the right-arrow button to move the selected tables to the right side.

   You can move a table back to the left side by selecting it and clicking the left-arrow button.
3. Optionally, if you want to add all columns from tables even if they are already displayed in the Grid panel, uncheck the Skip Duplicate Columns check box.
4. Click OK.

   Columns from the tables you selected are added to the bottom of the Grid panel listing. The view editor automatically creates aliases for identical column names, whether they came from the same table or different tables. You can select any alias on the Grid panel and type a new alias for the column.
5. Select the SQL panel and see that the `SELECT` statement now includes all of the columns you requested.

Adding an Individual Column

You can add individual columns from a table, and then move it to the position you want on the Grid panel.

To add an individual column as a row in the Grid panel

1. Click the Grid tab to open the Grid panel.
2. Click an empty cell in the left column (the Column column).
3. When the cell becomes a bordered field, click the drop-down button on its right and select a table column from the list.

   The list includes all columns from all tables in the model.
4. Optionally, clear the check box in the Output column to exclude the item from the view execution results.

Note: You can check or uncheck all check boxes in the Grid Output column at once, using the Select All Outputs or Clear All Outputs toolbar icons, respectively.

Creating an Alias for a Column

You can supply an alias that is the column name that users see when they browse or query the TDV data services you create, so it is wise to assign an alias that makes sense to those users.

Note: If you specify a reserved word as an alias, it is enclosed in double quotes in the SQL statement. For a list of reserved words, see the *TDV Reference Guide*.

To assign an alias to a column in the Grid panel

1. Click the cell under Alias that corresponds to the column item you want to give an alias.

   The editor may have assigned an alias automatically. You can highlight it and type a new alias to replace it.

2. Type an alias for this column in the cell.

Note: Automatically generated SQL from some ODBC clients (Excel and others) needs to be modified if you intend to use the alias to query the data sources.

Changing a Column Data Type using the CAST Function

Changing data types is sometimes necessary because the view might need to be consumed through a client interface that does not support that data type, or perhaps the data type defined in the data source is not optimal for your current needs.

To declare a variable in a SELECT statement

1. Open any view.
2. Select the Grid tab.
3. Select any row with data.
4. Right click and select Function > Convert > CAST.
5. Click OK.

Including a Column in the ORDER BY Clause

In the Grid panel you can add an `ORDER BY` clause to the SQL statement to sort the results by one or more columns, in ascending or descending order.

Note: If you choose sort order before you choose sort type, Unsorted is the only Sort Type option available.

You should not use OFFSET and FETCH in a TDV view.

To include a column in the ORDER BY clause

1. Click the cell in the Sort Order column for the row, and select a number to indicate the order in which that column is to be sorted (1 for first, 2 for second, and so on).

   Note: As you designate more rows, more sort-order numbers become available. If you specify the same number more than once, the editor assigns a distinct sort order number for the other row. Check that the assignments are what you want.

2. Click the cell in the Sort Type column for the row, and select Ascending, Descending, or Unsorted.

   Note: If you want to remove a row from the ORDER BY clause, select Unsorted in the Sort By column.

3. Click the SQL tab to look at the resulting ORDER BY clause.

Specifying a GROUP BY Clause

In the Grid panel you can group by an item, an expression, or a restriction involving aggregate functions.

To specify a GROUP BY clause

1. To group by an item, click the cell where the item's row crosses the Group By column and choose one of these options from the drop-down list:
   - None—to omit any GROUP BY clause.
   - Group By—to add a clause to GROUP BY the item in this row.
   - Expression—to group by an expression.
   - Having—to place restrictions (involving aggregate functions) on the rows returned from the GROUP BY clause.

Specifying Criteria on a Column

You can combine multiple criteria in one row using OR logic. Criteria entered in different rows (vertically) are combined using AND logic. A column does not have to be included in the result set to have criteria associated with it.

For example, in an Order table with a total_price column, if you want to retrieve large and small orders, you might enter `>100` in the Criteria column, and `<10` in the first Or column.

To specify criteria on a column

1. To specify a criterion on an item, click the cell where the item's row crosses the Criteria column and enter the first criterion in the cell.
2. To specify more criteria on an item (up to a total of four) click the cell where the item's row crosses an *Or* column and enter the criterion in the cell.

Including a Function in a SELECT or WHERE Clause

You can specify a function to include in a SELECT statement using a cell in the Column column for the item's row. You can specify a function to include in the WHERE clause of the SQL statement using a cell in the Criteria column for the item's row. The functions available are listed in "TDV Support for SQL Functions" in the TDV Reference Guide.

Note: The functions you can add include custom functions. See Promoting Procedures to Custom Functions, page 273 for a description of how administrators can promote procedures, which then appear in the drop-down list of functions.

To include a function in a SELECT statement

1. Right-click the cell in the Column section, and select Function > <function type> > <function name>.
2. Specify the input arguments for the function, and click OK.

   The function format is provided in the Function Arguments Input window. The function added in this way is included in the SELECT statement.

To include a function in a WHERE clause

1. Right-click the cell in the Criteria section, and select Function > <function type> > <function name>.
2. Specify the input arguments for the function, and click OK.

   The function format is provided in the Function Arguments Input window. The function added in this way is included in the WHERE clause.

Declaring a Variable in a SELECT Statement

The variables defined using the following steps are considered "virtual columns," and they are included in the SELECT statement. Virtual columns help to integrate views and procedures.

Typically, virtual columns are implemented in the following ways:

- As a range of values in the filter criteria
- As a single value so it can be used to call a procedure

Note: XML types are allowed. If you want to specify an XML schema, you must edit the SQL of the view directly and the declaration of your column needs to include the XML schema definition of that value. See the XML example that follows the steps.

To declare a variable in a SELECT statement

1. Right-click a Column cell, and select Declaration.
2. In the Add Declaration window, in the Parameter Name field, supply a unique name for the variable.
3. Specify the type of the variable in the drop-down list.
4. In the Default Value field, specify the default value for the variable.
5. Click OK.

The value you defined should appear in the Column cell in the form:

```
{DECLARE <variable_column_name> <data_type> DEFAULT
<default_value>}
```

Example of How to Modify the Declaration of the XML Virtual Column

To specify an XML schema, you must edit the SQL of the view directly and the declaration of your column needs to include the parts of the XML schema definition that are relevant for that value. After you edit the SQL of the view directly, you will not be able to edit your SQL model. The following shows the XML schema definition associated with a single virtual column:

```
{DECLARE Request_decl
XML('{http://www.compositesw.com/example/transaction/orders/v1.0}R
equest', '<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://www.compositesw.com/example/transaction/request/
v1.0" attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://www.compositesw.com/example/transaction/re
quest/v1.0"> <xs:element name="Request">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ReqId" nillable="true" type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>') DEFAULT NULL
}
{DECLARE inputXML_decl XML DEFAULT null} inputXML,
...
```

Example that Uses Procedures

This example uses two views and the procedure that is distributed with the product.

requires_prod_id View

```
-- This view requires that a single product ID to be specified, so
it can be used to join with
-- a lookup procedure.
SELECT
    a.ProductID,
    { DECLARE a_prod_id INTEGER } a_prod_id,
    a.ProductName,
    b.ProductDescription
FROM
    /shared/examples/ds_inventory/tutorial/products a INNER JOIN
        /shared/examples/LookupProduct(a_prod_id) b
    ON a.ProductID = a_prod_id
```

wrap_prod_id View

```
SELECT
    ProductID,
    a_prod_id,
    ProductName,
    ProductDescription
FROM
    /shared/DEMO/VirtualColumn/for_procedures/requires_prod_id
WHERE
    a_prod_id = 10
```

Example that Uses a Range of Values

This example uses two views and a procedure.

proc_wrap_prod_id Procedure

```
PROCEDURE proc_wrap_prod_id(
    IN prod_id_begin INTEGER,
    IN prod_id_end INTEGER,
    OUT result CURSOR (
        OrderID INTEGER,
        ProductID INTEGER,
        Discount NUMERIC(12,2),
        OrderDate DATE,
        CompanyName VARCHAR(50),
        CustomerContactFirstName VARCHAR(30),
        CustomerContactLastName VARCHAR(50),
        CustomerContactPhone VARCHAR(30)
        )
    )
    BEGIN
        OPEN result FOR
            SELECT
        OrderID,
```

```
        ProductID,
        Discount,
        OrderDate,
        CompanyName,
        CustomerContactFirstName,
        CustomerContactLastName,
        CustomerContactPhone
            FROM
                /shared/examples/ViewOrder ViewOrder
        WHERE
(ProductID >= prod_id_begin)
and (ProductID <= prod_id_end)
;

    END
```

requires_prod_id View

```
SELECT
products.ProductID,
{ DECLARE prod_id_begin INTEGER } prod_id_begin,
{ DECLARE prod_id_end INTEGER } prod_id_end,
products.ProductName
FROM /shared/examples/ds_inventory/tutorial/products products
WHERE
(ProductID >= prod_id_begin)
and (ProductID <= prod_id_end)
```

view_wrap_prod_id View

```
SELECT
    ProductID,
    ProductName
FROM
    /shared/DEMO/VirtualColumn/range_of_values/requires_prod_id
where
prod_id_begin = 5
and prod_id_end = 15
```

Designing SQL for a View in the SQL Panel

You can use the SQL panel in the view editor to type and edit a view's SQL. You can upload an existing SQL statement to use as a starting point using the Insert from File button.

Note: When you edit the SQL in the SQL panel, the design in the Model panel becomes invalid and the Model and Grid panels disappear. You can regenerate the model to redisplay the Model and Grid panels as described in Generating a Model from the SQL Panel, page 241.

There are some limitations and considerations when editing the SQL:

- The SQL for a view cannot contain an INSERT, UPDATE, or DELETE clause. You have to include such clauses in a SQL script, as described in Procedures, page 267
- If you want to use the Model panel later for the current view, first save the SQL statement under a different name, and then use the SQL panel for your hand-typed SQL code.
- You are responsible for the syntax of the SQL you type in the SQL panel. Studio does not check the validity of the syntax.
- If table or column names contain special characters, such as @, $, or &, enclose the names in double quotation marks (`" "`) in the query.
- You should not use OFFSET and FETCH in a composite view.

For details on the SQL features supported in TDV, see the *TDV Reference Guide*.

To create or edit a view's SQL in the SQL panel

1. Click the SQL tab in the view editor.
2. Type the SQL code for your view.

   You can do most standard editing tasks in the SQL panel. For a list of keyboard shortcuts, see Studio Code Editing Keyboard Shortcuts, page 660.
3. Save the view.

Note: You can also save the current SQL using the Save to File button.

Defining Primary Key for a View or Table in the Indexes Panel

In Studio, the Indexes panel in the view and table editors enables creation of metadata labels for existing indexes and primary keys already present in the data source. You can mark a column in a view as indexed or as a primary key.

Primary keys are similar to indexes. A column designated as a primary key signifies that every value in that column is unique. So when a column is made a primary key, not only is an index created for it, but typically the native data source ensures that every value in that column is unique and ensures this uniqueness for every update and insert.

Identifying indexes and primary keys enable the TDV Query Engine to use logical algorithms for faster more efficient joins that leverage organization of the data source.

To define and publish an index

1. Open a view or table editor, and click the Indexes tab.
2. Click Add.
3. In the New Index window, supply a name for the index, and click OK.

   The index is listed along with the column projections in the view, as shown in the next screen.
4. Select the column for the index and click the right arrow button.

   You can add as many columns as you want. When you add columns to the index, you can see the synchronization between the list of available columns and indexed columns.
5. Select Unique or Primary Key box to indicate the type of the index.

   If you select Primary Key, the Unique box is automatically selected.

   A Unique index is used when you do not want duplicate values to be retrieved. The same is true of Primary indexes. However, in a table you can have only one index marked as Primary key, but you can have more than one index marked as Unique.
6. (optional) Use the Add button to mark more columns as indexes, and use the Remove button to delete any columns improperly marked as existing indexes.
7. Save the resource.
8. Publish the indexed resource to make the index available for external clients.
9. Open the system table ALL_INDEXES (in Data Services/Databases/system).
10. Click Show Contents in the Columns tab to view the details of all the available indexes in the system.

    If you have published data source tables, the indexes on those resources would also be listed in ALL_INDEXES.
11. Click a row in the Result panel output to view the result details. For some extremely large results, such as those generated by using EXTEND on an array to generate BIGINT number values, some of the data might be truncated without a visual indication by Studio.

Defining a Foreign Key for a View or Table Resource

In a relational table, a foreign key is a column that matches the primary key column in another table. Suppose that column X in Table A relates to column Y in Table B and column Y is a primary key in Table B, the foreign key appears in Table A.

Foreign key relationships indicate that if you join A.X = B.Y, there is exactly one row in Table B found for each row in Table A. This information is a useful hint for join operations.

In Studio, the Foreign Keys panel in the view and table editor lets you create a definition that acknowledges and allows use of any foreign keys in the data source. For each foreign key you want to define, first you need to select the column that would be the foreign key, and subsequently identify the corresponding primary key column in the parent table.

The processes for defining a foreign key and testing the creation of a foreign key are as follows:

To define a foreign key

1. Open a view or table.

   The view used here has two tables—orderdetails and orders—from the data source ds_orders (/shared/examples/ds_orders).

   The projected columns are orderdetails.ProductID, orderdetails.UnitPrice, and orders.ShippingMethodID. See the Grid panel and Columns panels in the view editor.

2. Click the Foreign Keys tab in the editor to open the Foreign Keys panel.
3. Click the Add button (in the bottom left section of the editor).
4. In the input window that opens, type a name (for example, `FK_ProductID`) for the foreign key that you want to define, and click OK.

   The columns that are available for you to mark up as a "foreign key column" are listed in the section Available Columns. These columns are exactly the same columns selected for projection in the view or table. See the Columns panel of the editor.

5. Select a column that you want to identify as a foreign key.

   In this example, ProductID is selected.

   Specify the parent table that contains the primary for this foreign key column. The parent table is also known as the referenced table.

6. Click the Browse button, and locate the parent table in which ProductID is the primary key.

   In this example, the parent table is:
   `/shared/examples/ds_inventory/products`

7. Use the forward arrow button to move ProductID from Available Columns to the Foreign Column section.

The columns in the Parent Table are displayed in a drop-down list in the Primary Column section, and the primary key column is visible.

8. Select the primary key column (ProductID) in the drop-down list (in the Primary Column section) and save the resource. Now, the view has a column (ProductID) identified as a foreign key.

Repeating this process, you can define as many foreign keys as you need.

To test the creation of the foreign key

1. Publish the resource. For details on publishing, see Publishing Resources, page 407. You can also verify that the foreign keys you defined are accessible to external client applications.
2. Open the system table /services/databases/system/ALL_FOREIGN_KEYS, and view its contents.
3. Right-click on:

```
Data Services/Databases/system/ALL_FOREIGN_KEYS
```

4. Select Show Contents.

Or, open Data Services/Databases/system/ALL_FOREIGN_KEYS, and click Show Contents. The foreign keys you defined are listed in the Result panel.

Generating a Model from the SQL Panel

If you have typed or changed the SQL statement in the SQL panel, you can generate a model based on the SQL currently displayed in that panel. The Model and Grid panels reappear in the view editor when you generate a model.

The model generator does not support all of TDV SQL syntax. SQL features the model generator does *not* support are as follows:

- UNION, INTERSECT, EXCEPT, EXISTS, scalar subqueries, derived tables, IN clause with a subquery, quantified subquery, and the INSERT, UPDATE, and DELETE operations.
- Error messages result for models generated from queries that include these SQL features.

In the regenerated model, the tables would be joined at the top (joining the table tiles), instead of at the appropriate columns, under the following circumstances:

- If the ON clause of the JOIN has one or more of the following items:
  - A function
  - an OR condition
  - Any of the following predicates: IN, LIKE, BETWEEN, IS NULL
- If the join is a self-join and no columns are involved in the join, or only columns from the same table are involved in the join.

The model generator accepts all INNER and OUTER JOINs. However, it might not be able to match the columns originating from the left and right sides of the join. When the model generator cannot identify both columns, the two tables in the resulting model are joined by a line that stretches from the title bar of the first table to the title bar of the second table.

To generate a model from the SQL panel

1. Click the Generate Model toolbar button in the view editor's SQL panel.

   Note: If Studio generated the SQL, the Generate Model toolbar button is not available.

Working with Views and JSON

JSON files are widely used to collect and transfer data, especially through the web. The structure of a JSON file can readily be converted into a structure that resembles a table. The 'virtual table' can be inserted into an existing database table, or it can be queried using a SQL JOIN expression. After creating a view for your JSON file, you can then reference and use the resulting JSON table in any other view, or you can use JSON_TABLE syntax to define the JSON table structure and populate it from within you view.

TDV and Studio support the use of the SQL JSON_TABLE function. For more information about the SQL syntax, see the *TDV Reference Guide*.

You can use JSON_TABLE syntax in various ways within TDV. The following sections describe two common scenarios:

- Using a Referenced JSON File within a TDV View, page 243
- Using a JSON Table Within a TDV View, page 243
- JSON Table Examples, page 244

Using a Referenced JSON File within a TDV View

When using the JSON_TABLE syntax to reference a JSON file, you can have TDV pull the data from the file and have the view define the table structure that you want to use for the data.

To add and use a referenced JSON file within a TDV View

1. Make sure that your data source has been added to Studio and that the JSON file with the data that you want in the view is present within Studio.
2. Add a view to describe the JSON file using the JSON_TABLE SQL function.
3. Add another view. From within this view you can reference your view with the JSON table and use standard SQL functions to manipulate the composite view.

Using a JSON Table Within a TDV View

When using the JSON_TABLE syntax in a view, you can define the structure and the data.

To add and use a JSON table within a TDV View

1. Add the JSON_TABLE SQL function with the data and structure definitions to the SQL tab of your view.

   For example, the JSON table could have:

```
FROM JSON_TABLE('{
    "company": {
        "department": [
            {
                "DepartmentID": 1,
                "DepartmentName"  :  "Sales"

            },
            {
                 "DepartmentID": 2,
                "DepartmentName"  :  "Project"
            },
            { "DepartmentID": 3,
              "DepartmentName"  :  "Market"
            },
            { "DepartmentID": 4,
              "DepartmentName"  :  "HR"
            }
        ]
    }}',
'$.company.department'
```

2. From within this view you can now use standard SQL functions to manipulate the composite view.

JSON Table Examples

For example, the SQL for your entire view might resemble one or more of the following.

View with a Literal JSON Table

```
SELECT
columnName,
columnValue
FROM JSON_TABLE('{
"company": {"department": [
{ "DepartmentID": 1, "DepartmentName" : "Sales" },
{ "DepartmentID": 2, "DepartmentName" : "Project" },
{ "DepartmentID": 3, "DepartmentName" : "Market" },
{ "DepartmentID": 4, "DepartmentName" : "HR??" },
{ "DepartmentID": 5, "DepartmentName" : "Service" },
{ "DepartmentID": 6, "DepartmentName" : "Advertisement" }]
}}',
'$.company.department'
COLUMNS(columnName VARCHAR(20) PATH KEY,columnValue VARCHAR(100)
PATH VALUE)) JT
```

View with a JSON Table and a Self-Join

```
SELECT
    two_references.customerId, two_references_1.price
FROM
    /shared/myViews/json/dynamic/reference_list/two_references
two_references INNER JOIN
        /shared/myViews/json/dynamic/reference_list/two_references
two_references_1
    ON two_references.customerId = two_references_1.customerId
```

View with JSON Table and Inner Join

```
SELECT
    literal_json_table_B.employeeName,
    literal_json_table_A.departmentID,
    literal_json_table_A.departmentName
FROM /shared/jsontable/literal_json_table_B inner JOIN
        /shared/jsontable/literal_json_table_A
    ON literal_json_table_B.departmentID =
literal_json_table_A.departmentID
order by literal_json_table_B.employeeName
```

Designing Column Projections Using the Columns Panel

If an external client must receive the data according to a particular external schema with specific data type projections, you can use the Design Mode check box in the Columns panel. The Design Mode check box lets you design a projection of a view, so that you can do a top-down design from the SQL panel.

When you use the Columns panel to design the projections, the SQL implementation is not automatically updated in the SQL panel. You must make sure there is a match between the columns defined in the SELECT statement of the SQL and columns you designed in the Columns panel using the design mode, including the order in which they are provided; otherwise, the query will generate an error when executed.

To design column projections through the Columns panel

1. Open the Columns Panel in the view editor.
2. Check the Design Mode check box in the upper right corner.

   Studio displays the warning below to let you know that changes you make to the Columns panel are not automatically made in the SQL panel which invalidates the SQL implementation; you must make the same edits in the SQL panel to ensure that the query works.
3. Highlight the row below which you want to add the column projection.
4. Click the Add button, and select a data type from the drop-down list.

   A new row is created, with a default name and the selected data type for the new column projection.
5. Optionally, highlight the default column name and type a new column name.
6. If the data type needs to change, right-click the Type/Reference value for that row, select Change Type, and specify the correct data type for the column projection.
7. Edit the SQL to include your columns in the `SELECT` statement.

After you have specified all columns are specified in their result-set display order, you can use them in the SQL statement in the SQL panel. However, the SQL is not automatically updated to reflect the design you created in the Columns panel.

Obtaining View Details

You can find out the details about a resource, such as its name, location in the resource tree, type and owner, and whether it is locked or not, using the Info tab.

To obtain the details about a view

1. Click the Info tab in the view editor.
2. Annotations are shown in the tooltooltipplayed when the mouse cursor pauses over a resource in the Studio resource tree, helping identify what the resource represents.

Executing a View

After you have designed and saved a view (refer to Designing a View and Table Resource, page 220) you can execute its SQL from within Studio to see the result.

For executing a view's SQL through client applications, see *TDV Client Interfaces Guide*.

Executing a view might be affected if the view uses a data source that was originally added to TDV Server using the pass-through mode without saving the password. To execute such a view, you must log into TDV Server with the same logisign-indentials (userusername password) that are necessary to log into the data source. For further information on pass-through credentials, see the details for the Save Password and Pass-through Login fields under Adding a Data Source, page 68.

Executing a view from Studio is blocked if the view's SQL is SELECT * on a view that includes a table with column-based security (see "About Managing Dependency Privileges" in the *TDV Administration Guide*).

To execute a view from Studio

1. Double-click the view to open it, or right-click the view. Select Open.
2. Click the Execute button.

   The Result panel displays the first 50 rows (default) of view execution results.
3. Use the Load More Results button to view the next 50 rows.

Generating a Query Execution (Explain) Plan

You can show the query execution plan instead of executing the query.

To show a query execution plan from Studio

1. Open a view.
2. Click the Show Execution Plan button.

   The Execution Plan panel displays the query execution plan.
3. Scroll down to see more of the query execution plan.

Generating an Explain Plan and Displaying it in a Client Application

An alternate to show a query execution plan is to precede the query with the keyword EXPLAIN. This feature can be used in Studio, although it is intended for use in JDBC/ODBC client applications. Being able to analyze the execution plans of a query in tools other than TDV can help you optimize your queries faster. The text results are retrieved in a result set that can be consumed by Studio or by your client application. The result set is one column of VARCHAR text.

The default column width of the execution plan is 100 characters. You can change it from Studio using the TDV Explain text width configuration parameter.

Option	Description	Example Syntax
show_source_plan="true"	Retrieves the query plan. This can also be used in the SQL Scratchpad.	`explain select {option show_source_plan="true"} * from <view>`
show_runtime="true"	Retrieves the execution statistics (plan and runtime statistics). This can also be used in the SQL Scratchpad.	`explain select {option show_runtime="true"} * from <view>`

To explain a query execution plan within your client application

1. Use your client application to connect through JDBC or ODBC to TDV.
2. Navigate to the TDV table or view with the query you want to analyze.
3. In the SQL editor for that query, type EXPLAIN before the query. For example:

```
explain select * from emp;
```

4. Run the query.
5. Review the explain plan to determine how to optimize the query.

To explain a query execution plan within Studio

1. Open a view.
2. In the SQL Scratchpad, add the keyword EXPLAIN before the query.
3. Click the Execute button.

   The Result panel displays the view's query execution plan (rather than the view's execution results). The actual view is not executed.
4. Use the Load More Results button to view the next 50 rows of the plan.

Rebinding a View

A view depends on one or more underlying sources, and the view is considered bound to those underlying sources. A view can be bound to tabular data or a procedure. Rebinding is useful when:

- You create a view with its sources and later decide to rebind the view to different sources.
- An execution error has occurred for a view because a source with which the view was initially bound does not exist any more.

To rebind a view to a different source

1. Open the view.
2. Click the Show Rebind Panel button on the Studio editor toolbar.

   The Rebind panel opens in the lower part of the view editor panel and displays the dependencies as determined the last time it was saved.

   If a view has never been saved, the rebind window is empty.
3. In the Rebind panel, select the source from the list in the left pane.
4. Click the Rebind button.

   The Rebind window appears, displaying the resources available for binding.
5. In the Rebind window, specify the source with which you want to rebind the view, and click OK.

6. Save the view.

If you modify the view or its dependencies, the binding is automatically updated when you save the view, and the Rebind panel reflects the change. For example, if your view has a data source dependency named ds_orders, and you rename it ds_orders_renamed and save the view, the Rebind panel lists the dependency as ds_orders_renamed.

For details on a view's lineage and dependencies, see Displaying the Lineage of a View, page 249.

Displaying the Lineage of a View

When you are working on a complex query, it is useful to know what resources are involved so you can understand the query's relationships to other resources. The relationships displayed include the resources on which the view depends and any resources that depend on the view. The Lineage panel in Studio shows a view's lineage. The Lineage panel is also available for cached views as well as other resource types. See Exploring Data Lineage, page 445, for more information.

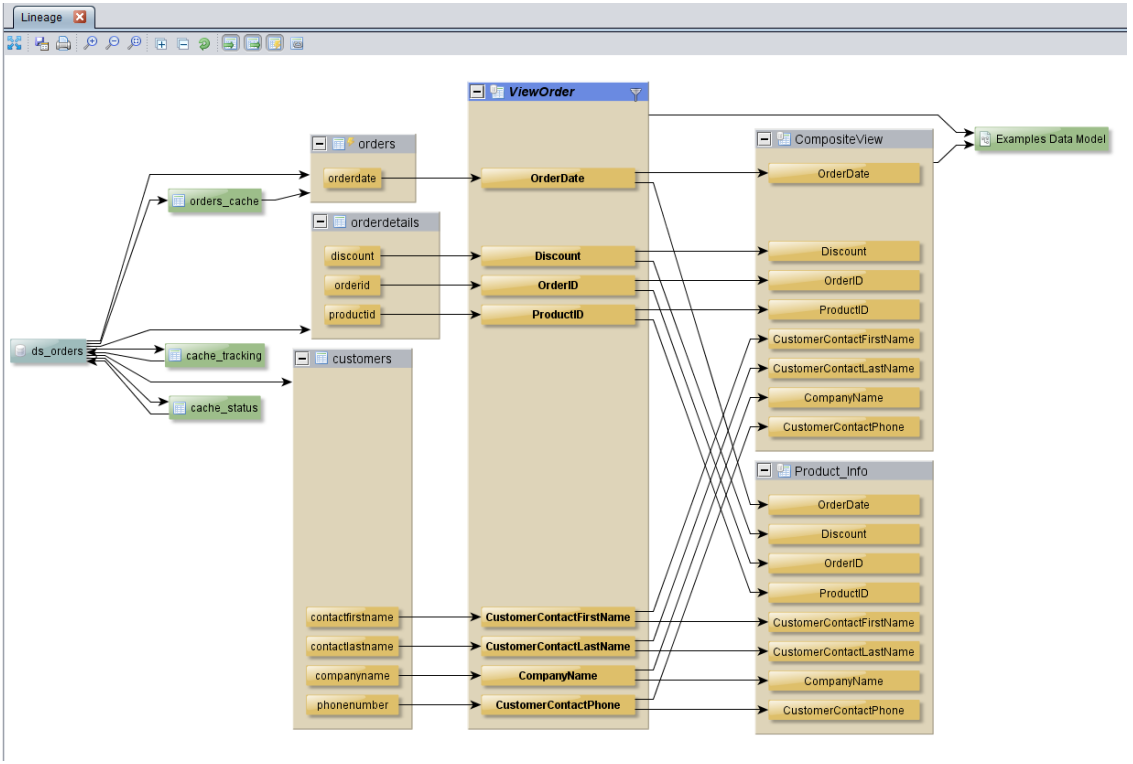
To display the lineage of a view

1. Open the view and click the Open Lineage Panel toolbar button. For OLAP Views this button is on the SQL tab of the editor.

   Note: You can also display the lineage for the view by right-clicking a view in the resource tree, and selecting Open Lineage. A new tab opens in the Studio workspace. This method of displaying the view lineage provides more space and is useful if your view has many resources.

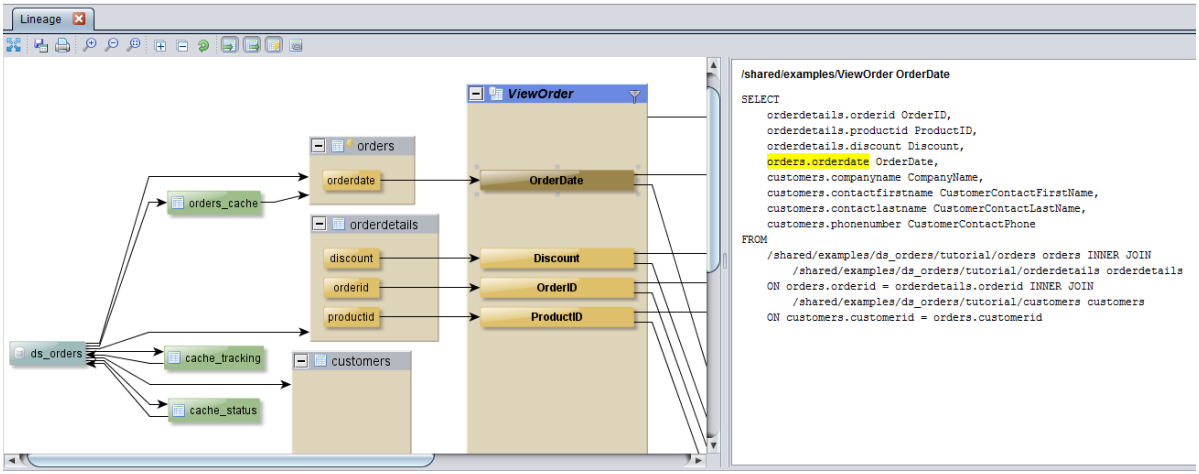
   The Lineage panel opens in the lower section of the view's editor and displays all the resources involved with the view in a graphical format. The following

screen shows the resources (data sources and tables) on which the sample view ViewOrder depends and the resources that depend on this view.



In the graphical format, the view's dependencies are shown on its left and the resources that depend on it are shown on its right.

2. Select any resource in the Lineage panel to see details about the resource in the right pane. For example, select a table in a view to display the SQL used to generate that view with the selected table highlighted.



3. Use the buttons at the top of the Lineage panel to adjust the display. See Lineage Panel Buttons and Controls, page 447 for more information.

See Working with the Data Lineage Graph, page 450 for more information about the Lineage panel.

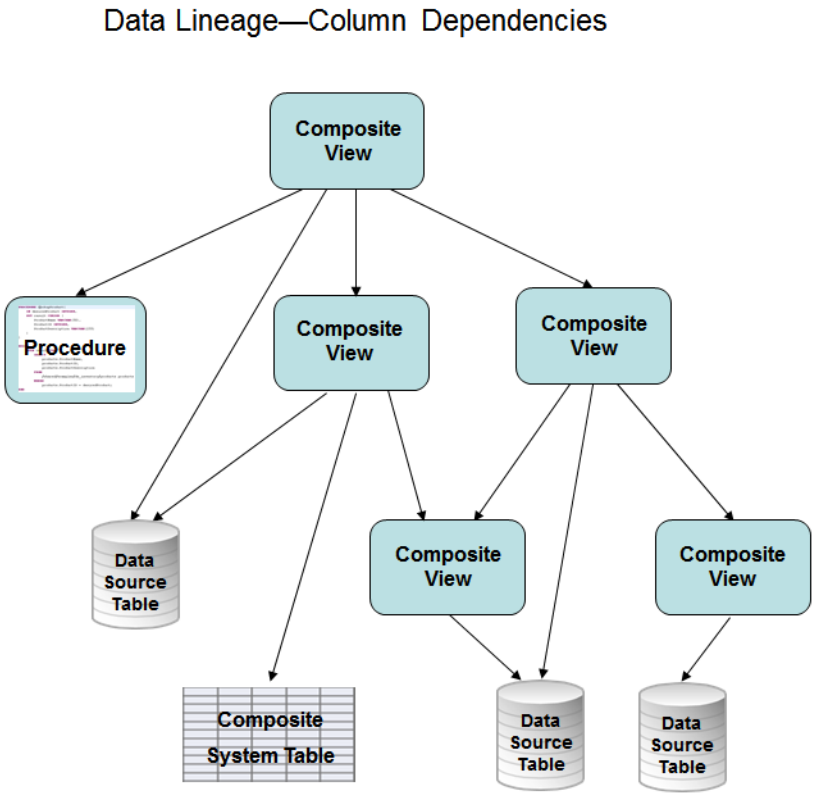
View Column Dependencies and References

TDV provides API tools for you to ascertain the data lineage of the resources involved in a query. You can determine:

- Column dependencies—How the columns are derived from a specified composite view and going back one or more levels through the various resources on which the view depends. You can find out what resources are involved and what transformations might have occurred that affect the results. See Getting Column Dependencies Using the API, page 252, for more information.
- Column references—What resources directly depend on that column. You can find out what other views refer to a specific column in a table or view. See Getting Column References Using the API, page 258, for more information.

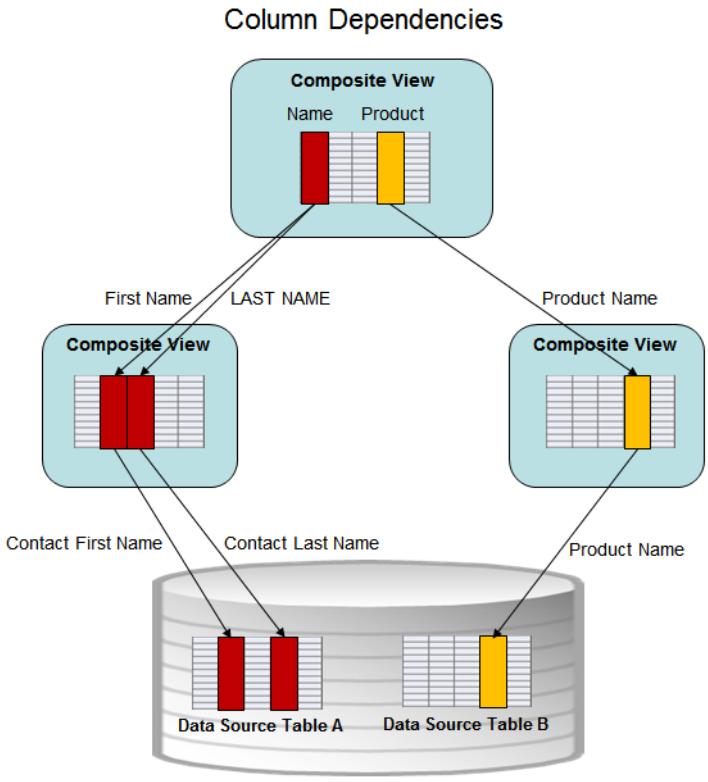
Getting Column Dependencies Using the API

TDV provides a system procedure that you can use against a specified composite view to get its column dependencies, discover where the data comes from, and see how it is derived. The data can come from many different resource types and paths including tables, views, procedures, packaged queries, delimited files, and Web service operations. Some of these possibilities are illustrated here:

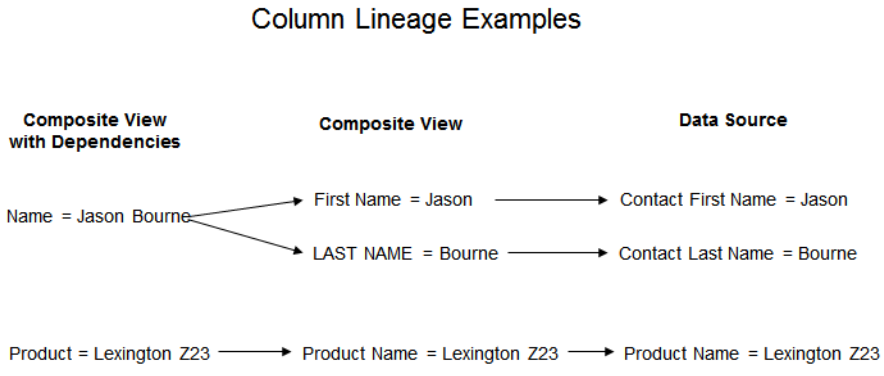


If the view depends on a procedure, web service, or packaged query, that is the end of the dependency analysis for that data. No further analysis is done through a procedure, web service, or packaged query.

After you request the column dependencies for a view, the GetColumnDependencies procedure results tell you, for each column in the view, what data source originated the data and what views transformed or filtered the data as shown in this example:



In this example, the column lineage data results might be as shown here:



The GetColumnDependencies procedure lets you:

- View the resources involved in creating column results going all the way back to the data source or procedure.
- Find out which columns have been transformed and how.
- Find out which columns are not transformed.
- Discover the original data source type.
- Find out which columns involve constants.
- See who originally created the resources involved and when.
- Create views on top of the procedure to analyze and aggregate data source information.
- Publish the GetColumnDependencies procedure:
  - As a new Data Service. You can then access it with the supported protocols like JDBC, ODBC, ADO.NET.
  - As a new Web service which you can access using REST, SOAP, and the other supported protocols.

To get the column dependencies for a view

1. In Studio, open the view containing the columns you want to learn about.
2. Click the Info tab and copy the full contents of the Name field which is the path for the view.
3. In the localhost directory, navigate to the lib/resource folder and open the GetColumnDependencies procedure.
4. Click Execute. Studio prompts you to enter the input parameters for this procedure.
5. Paste the view pathpathname copied earlier into the resourcePath field.
6. Enter the other input values for the procedure as shown in the table.

Parameter	Value Description
resourcePath	Required Enter the path of the resource to be analyzed. The supported resource types are TDV SQL Views in plain or published form.

Parameter	Value Description
columnFilter	Optional. Specify if the column lineage results should be filtered. If empty, no filtering occurs. To filter the results, enter a comma-separated sequence of case-insensitive column names, indicating the columns whose dependencies should be analyzed.
ignoreCaches	Optional. Specify if the analysis should ignore whether depended resources are cached or not. Enter one of these values: true—Do not return any caches as part of the column lineage. false—The default. If blank or false, any existing caches in the column lineage are returned in the procedure results.
recursively	Optional. Specify if the analysis should be done recursively all the way to the source level of dependency or only return one level of dependency. Enter one of these values: true—Return all levels of dependencies down to the original source level. false—The default. If blank or false, return only a single level of column dependency.

The analysis is done based on the view definition, independently of whether the view is cached or not.

7. Click OK.

   TDV executes the procedure and displays the column dependencies in the lower Result For columnDependencies panel.

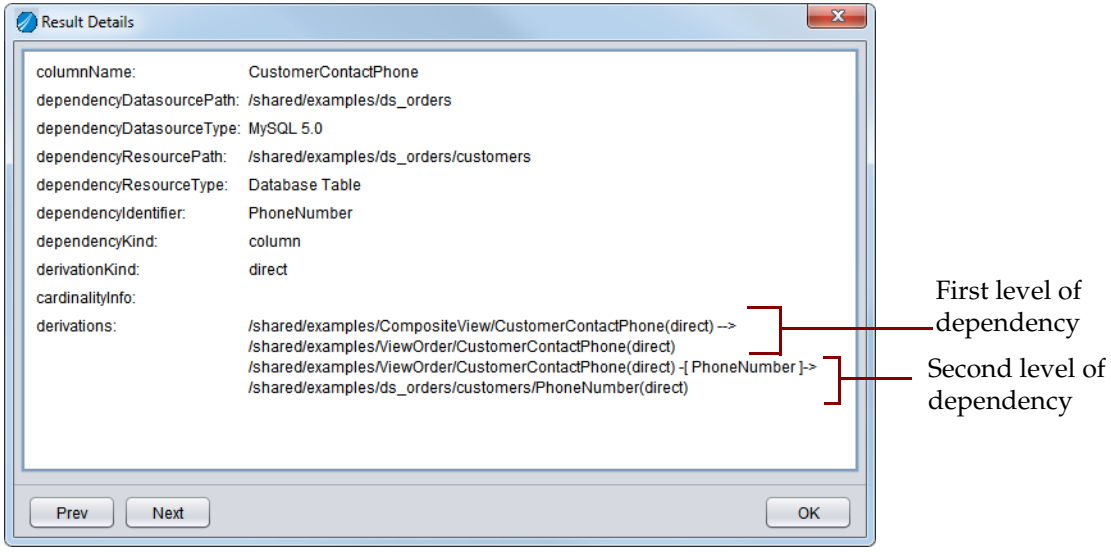
   The contents of the columns are described below:

Column	Description
columnName	The name of the resource column having the column dependency encoded in the row.
dependencyDatasourcePath	The path to the data source containing the resource owning the dependency. Empty if not applicable.
dependencyDatasourceType	The type of the data source containing the resource owning the dependency. Empty if not applicable.
dependencyResourcePath	The path to the resource owning the dependency. Empty if not applicable.

Column	Description
dependencyResourceType	The type of the resource that owns the dependency. Empty if not applicable. The set of data source types consists of all the data source adapter names accepted by TDV. The table and procedure resource types accepted by TDV is as follows: Database Table Delimited File Excel Table TDV SQL View System Table SAP RT Table SAP RFC Table SAP AQ Query Table SAP Infoset Query Table Siebel Table Database Stored Procedure Packaged Query Java Procedure Web Service Operation Composite SQL Script Procedure XQuery Procedure Transform Procedure Basic Transform Procedure Stream Transform Procedure XQuery Transform Procedure XSLT Transform Procedure
dependencyIdentifier	The column name if the dependency is on a column; otherwise, a literal.
dependencyKind	One of the following: • column, to indicate a dependency on a column. • literal, to indicate a dependency on a constant value. • parameter, to indicate a dependency on a dynamic value provided at runtime.

Column	Description
derivationKind	One of the following: • direct, to indicate that the value of the dependency is preserved by the dependent column • indirect, to indicate that the value of the dependency is transformed by the dependent column
cardinalityInfo	When applicable, one of the following: • aggregate, to indicate that an aggregate function is involved in the derivation of the dependent column • analytic, to indicate that an analytic function is involved in the derivation of the dependent column
derivations	When the dependent column is not a direct projection of the dependency, this field denotes how the dependent column is derived.

8. To view the details of the column dependency, select a dependency and click the Details button in the Result For columnDependencies panel. Studio displays the full details for the selected column dependency as shown in this example:



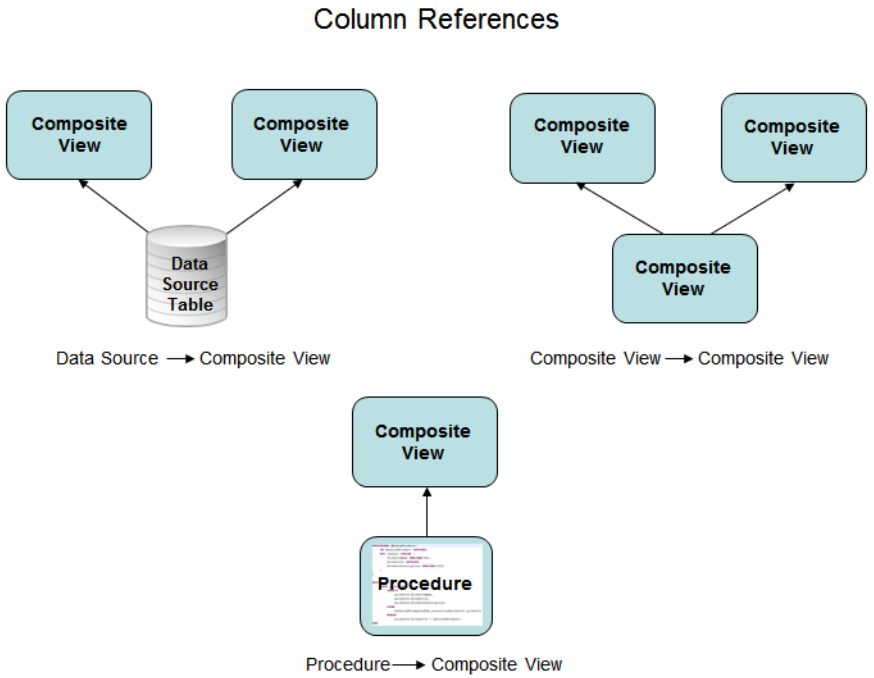
9. Optionally, save the results as a Comma-Separated Values (*.csv) file by clicking Save to File on the Result For columnDependencies panel.

10. Optionally, open any of the involved resources to see who originally created the resource and when on the resource Info tab.

Getting Column References Using the API

TDV provides a system procedure that you can use to get column references and discover what views and columns directly reference a specified view column.

This procedure can be executed on a data source table or composite view to understand what upstream references each column in the table or view has. Only one reference level at a time is returned, but you can run the GetColumnReferences procedure repeatedly to discover the path of the references.



The GetColumnReferences procedure lets you:

- List all views and columns that directly reference a view.
- See the type of reference which might be a SELECT clause, a FROM clause, a WHERE clause, or a TIMESERIES clause, or other type of reference.
- See if the column was transformed.
- See who originally created the resources involved and when.

- After you have determined the column reference, you can run the GetColumnReferences procedure again on the referenced view or table to determine the next level of reference.
- Publish the GetColumnReferences procedure:
  - As a new Data Service. You can then access it with the supported protocols like JDBC, ODBC, ADO.NET.
  - As a new Web service which you can access using REST, SOAP, and the other supported protocols.

To get the column references for a view

1. In Studio, open the data source table or view containing the columns you want to learn about.
2. Click the Info tab and copy the full contents of the Name field which is the path for the view.
3. In the <localhost> directory, navigate to the lib/resource folder and open the GetColumnReferences procedure.
4. Click Execute. Studio prompts you to enter the input parameters for this procedure.
5. Paste the table or view pathpathname copied earlier into the resourcePath field.
6. Enter the other input values for the procedure as follows:

Parameter	Value Description
resourcePath	Required. Enter the path of the resource to be analyzed. The supported resource types are TDV SQL Views or data source tables.
columnFilter	Optional. Specify if the column lineage results should be filtered. If empty, no filtering occurs. To filter the results, enter a comma-separated sequence of case-insensitive column names, indicating the columns whose references should be analyzed.

Note: The analysis is done ignoring whether the specified view or table is cached or not.

7. Click OK.

   TDV executes the procedure and displays the column references in the lower Result For columnReferences panel.

The contents of the columns are described in the following table.

Column	Description
columnName	The name of the resource column having the column reference encoded in the row.
referentResourcePath	The path to the resource containing the column reference. Empty if not applicable.
referentContext	The context in which the column is referenced: • In a WITH clause, to indicate a reference within a WITH clause. • In a SELECT clause as output, to indicate a reference that is projected by a SELECT clause. • In a SELECT clause as input, to indicate a reference that is used, but not projected, by a SELECT clause. • In a FROM clause, to indicate a reference within a FROM clause. • In a WHERE clause, to indicate a reference within a WHERE clause. • In a TIMESERIES clause, to indicate a reference within a TIMESERIES clause. • In a GROUP BY clause, to indicate a reference within a GROUP BY clause. • In a HAVING clause, to indicate a reference within a HAVING clause. • In an ORDER BY clause, to indicate a reference within a ORan ORDERclause.
referentColumnName	The name of the referent column. Populated only if referenceContext is in a SELECT clause as output; otherwise, empty.
derivationKind	Indicates the type of derivation: • Direct, to indicate that the value of the dependency is preserved by the dependent column. • Indirect, to indicate that the value of the dependency is transformed by the dependent column. Populated only if referenceContext is in a SELECT clause as output; otherwise, empty.

Column	Description
cardinalityInfo	When applicable, one of the following: • Aggregate, to indicate that an aggregate function is involved in the derivation of the dependent column. • Analytic, to indicate that an analytic function is involved in the derivation of the dependent column. Populated only if referenceContext is in a SELECT clause as output; otherwise, empty.
reference	A textual representation of the column reference.

8. To view the details of a column reference, select the reference and click the Details button in the Result For columnReferences panel. Studio displays the full details for the selected column reference.
9. Optionally, run the GetColumnReferences procedure again on the referenced view or table in the Result For columnReferences panel to determine the next level of reference.

   You can run the GetColumnReferences procedure as many times as needed to discover the reference path.
10. Optionally, save the results as a Comma-Separated Values (*.csv) file by clicking Save to File on the Result For columnReferences panel.
11. Optionally, open any of the involved resources to see who originally created the resource and when on the resource Info tab.

Creating a View from a Cube

These objects are only valid if you have the SAP BW adapter installed. For more information on defining and using this type of Studio resource, see the *TDV SAP BW Adapter Guide*.

Ad Hoc SQL and the SQL Scratchpad Editor

TDV provides an ad hoc SQL editor a SQL Scratchpad editor that let you execute "anonymous" SQL statements that are not bound to a resource. You can define and execute any SQL statements that can be executed for a composite view.

Ad hoc SQL content and history are saved on the local (Studio instance) machine, and retrieved in the following Studio session.

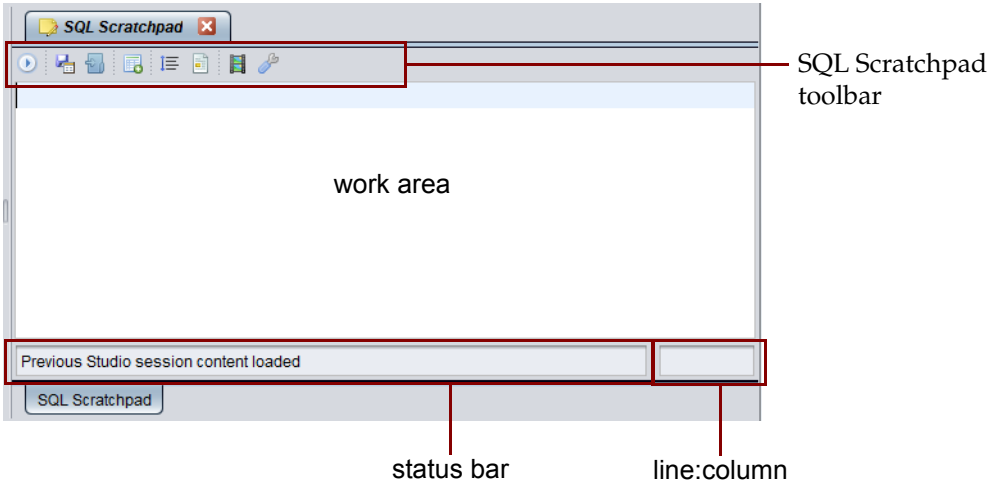
These topics are covered:

Opening the SQL Scratchpad Editor

There is only one instance of the SQL Scratchpad editor. If it is already open but not the active editor, any of the three methods listed below makes it the active editor by bringing it to the front.

You can open the SQL Scratchpad from the Studio Modeler panel in one of three ways:

- From the Studio menu bar, select File > Show SQL Scratchpad.
- On the Studio toolbar, click the Show SQL Scratchpad button.
- Press Ctrl+Q anywhere in the Studio window.



The short bar to the right of the status bar indicates the current line and column position of the cursor, separated by a colon. For example, 1:31 indicates that the cursor is on line 1, column 31.

You can type queries in the work area and add resources to the queries by:

- Typing the paths and names of the resources
- Dragging the resources from the Studio resource tree and dropping them onto the work area
- Using the Add Resources button

The following table describes the actions you can perform in the SQL Scratchpad Editor toolbar.

Label	Use to...
Execute	Execute the SQL query currently displayed in the panel and opens the Result panel (The SQL Scratchpad Result Panel, page 265) below it, showing up to 50 rows of results. Each time you execute a unique SQL query, it is moved to the top of the list in the History panel.
Export to File	Open a dialog box to save the SQL query as a file.
Import from File	Open a dialog box to import a SQL query from a file.
Add Resources	Open a window where you choose a resource to add to the SQL Scratchpad editor.
Format Query	Format the SQL text according to the settings made on the Editors tab of the Studio Options window.
Toggle Syntax Highlighting	Change between displaying all SQL text in the same font color (usually black) and displaying comments, keywords and literals in the colors you selected in the Editors tab of the Options dialog box.
Show/Hide History View	Display or hides a History pane along the right side of the SQL Scratchpad editor. (See The SQL Scratchpad History Pane, page 264).
Show Option Dialog	Open the Studio Options window to the SQL Scratchpad tab, where you can change the options for this window.

Composing and Executing an Ad Hoc SQL Query

This section describes how to create a new ad hoc SQL query in the SQL Scratchpad editor.

To create and execute an ad hoc SQL query

1. Open the SQL Scratchpad editor.

   See Opening the SQL Scratchpad Editor, page 262 for the three ways you can do this.

2. In the main scratchpad area, type your query.

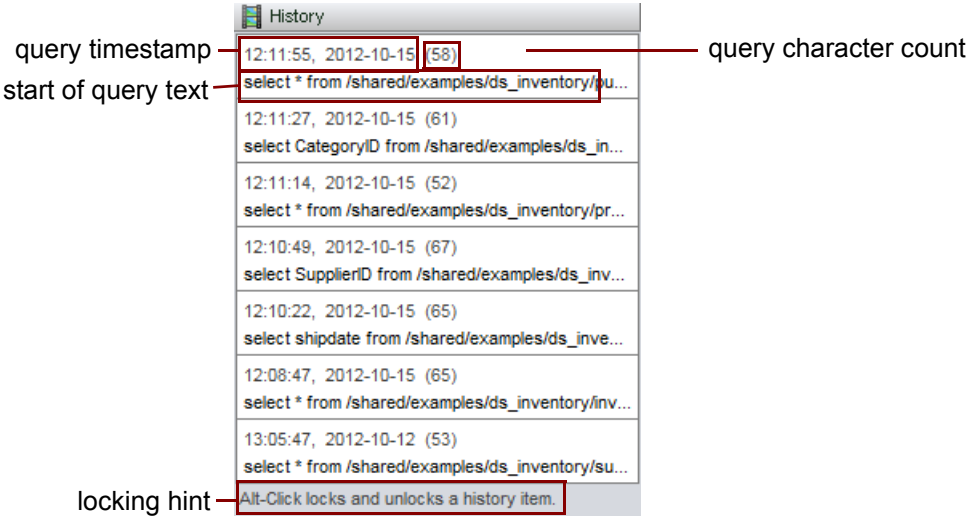
3. Click the Execute button.

After you click the Execute button, a Result panel (see The SQL Scratchpad Result Panel, page 265) appears and shows query execution in progress and the results returned. You can dismiss the Results panel by clicking the Delete button on the right side of its tab.

Note: If you precede the SQL query with the keyword EXPLAIN, the Result panel displays the query execution plan. In this case, the query is not executed; rather, the plan is formulated and displayed for the user to examine. This plan text is also available to JDBC and ODBC clients.

The SQL Scratchpad History Pane

If you click the Show History View button, a History pane appears on the right edge of the SQL Scratchpad editor, listing the SQL queries that you have created or changed in this window.



The History pane lets you:

- See the distinct queries you have executed. The history list retains the number of queries you specify in the Studio Options window, with the most recently used query at the top.
- Instantly display a previous query in the work area.
- Save all distinct queries automatically.
- Lock a query so it cannot be deleted.

  You can Alt-click an unlocked query in the history list to lock it. You can lock as many queries as you want.

  When a query is locked, a lock icon appears next to it. Locked queries do not count toward the Maximum History Size set in the Studio Options window.
- Unlock a query so it can be deleted. Alt-click a locked query in the history list to unlock it.
- Delete a selected query from the history list (after unlocking it, if necessary).
- Change the maximum number of queries to save in the history list. Click Maximum History Size in the Studio Options window and choose a number from the drop-down list.

  Changes to maximum history size do not take effect until you restart Studio.

The SQL Scratchpad Result Panel

The Result panel appears in the bottom area of the SQL Scratchpad editor when you click the Execute button on the toolbar. The Result panel displays up to 50 lines of execution results. If more than 50 lines were returned, you can see more of them by clicking the Load More Results button.

The table below describes the buttons and the actions you can perform in the Result panel toolbar. The Cancel button appears in the Result panel during execution of long queries, letting you halt execution. When you close the Result panel, any query execution in progress is canceled.

You can dismiss the Result panel by clicking the Delete button on its tab.

Label	Use to...
Save To File	Open a dialog box to save the results as a file. The default file type is a comma-separated values file (*.csv).
Details	Open a Result Details window that lists all of the column headings and values for the selected row. Click Next to list details for the next row, or Prev to list details for the previous row. To dismiss the window, click OK.

Label	Use to...
Load More Results	Load the next rows (up to 50) of the result. If 50 or fewer results were returned, this button is dimmed.
Clear	Clear all results from the Result panel.
Show Execution Plan	Open a tab, next to the Result tab, that shows the execution plan for the SQL executed.

If you click Show Execution Plan on the Result tab, an Execution Plan opens adjacent to the Results tab. The Execution Plan tab shows the execution plan and details for the SQL executed. The execution ID is shown on both tabs, so that the two can be correlated. An Execution Plan tab to the left of a Result tab indicates a stale plan.

Procedures

This topic describes how to design procedures to query and manipulate data stored in a data source. It also describes transformations and how to use them to map your data into different formats.

Transformations are a special class of procedures. For more information about working with transformations, see Using Transformations, page 311.

About Procedures

Procedures are predefined programs that can be executed to query and manipulate data stored in a data source. They have scalar input and output parameters. Some of them return one or more cursors, which represent tabular output data.

TDV supports the following types of procedures.

Procedure Type	Further Information
SQL scripts	Working with SQL Scripts, page 270
Custom Java procedures	Java Procedures, page 277

Procedure Type	Further Information
Packaged queries	Packaged Queries, page 284
Parameterized queries	Parameterized Queries, page 293
Physical stored procedures	Physical Stored Procedures, page 298
Transformations	Using Transformations, page 311

Although different languages (TDV SQL script, Java, SQL native to data sources, XSLT, XQuery) are used to formulate procedures, they all function alike in the TDV system. You can invoke one procedure type from within another.

You can use any text editor to write a procedure, and use Studio to add it to the server. You can also use Studio's procedure editor to create and edit any type of procedure, except Java procedures. You can add a procedure to any location except Data Services in the resource tree. For details, see Locating a Container for a TDV Resource, page 40.

For details on creating or adding procedures, see:

- Creating a SQL Script Procedure, page 268
- Adding a Custom Java Procedure, page 167
- Java Procedures, page 277
- Creating a Packaged Query, page 285
- Quick Steps for Creating a Parameterized Query, page 293
- Creating an XML, XSLT, or Streaming Transformation, page 314
- Creating an XQuery Transformation, page 320

For a description of how to move a procedure to a Data Services container, or if you want to make a procedure available to client programs or as a Web service, you must publish that procedure. For details, see Publishing Resources, page 407.

Creating a SQL Script Procedure

This section describes how to create a SQL script—a procedure written in TDV's SQL script language, as described in "TDV SQL Script" in the *TDV Reference Guide*.

The parameters you define in the SQL script panel must match the parameters that you define in the Parameters panel, including the order in which they are defined.

To create a SQL script

1. Right-click the container to which you want to add the script, and select New SQL Script.
2. Type a name for the script in the Input dialog, and click OK.

   The editor opens in the My SQL Script panel on the right.

   For information about the procedure editor, see:

   — Procedure Editor Panel Reference, page 664
3. Use the SQL Script panel to formulate and edit the SQL script.

   — You can use any non-declarative statement as the first statement after BEGIN.

   — The first non-declarative statement after BEGIN must not end with a semicolon (;), but every other statement must end with a semicolon.

   — If you have a script stored in your file system, you can upload it using the Insert from File button.

   — Type the script between the lines BEGIN and END.

   — The SQL Script panel can highlight SQL syntax elements (comments, keywords, and so on). You can turn highlighting on and off with the Toggle Syntax Highlighting toolbar button. To change highlighting, select Edit > Options in the main toolbar and open the Editors tab.

   — Drag and drop tables, views, or other procedures from the Studio resource tree into the procedure editor panel. This action inserts the full TDV resource name at the drop point.

   — Insert standard SQL or C-style comments by highlighting one or more lines and typing:

   — Ctrl-Hyphen to insert two dashes (--) at the start of each line.

   — Ctrl-/ (forward-slash) to enclose the lines between /* and */.

   See Commenting SQL, page 218 for information about inserting comments into SQL in TDV.

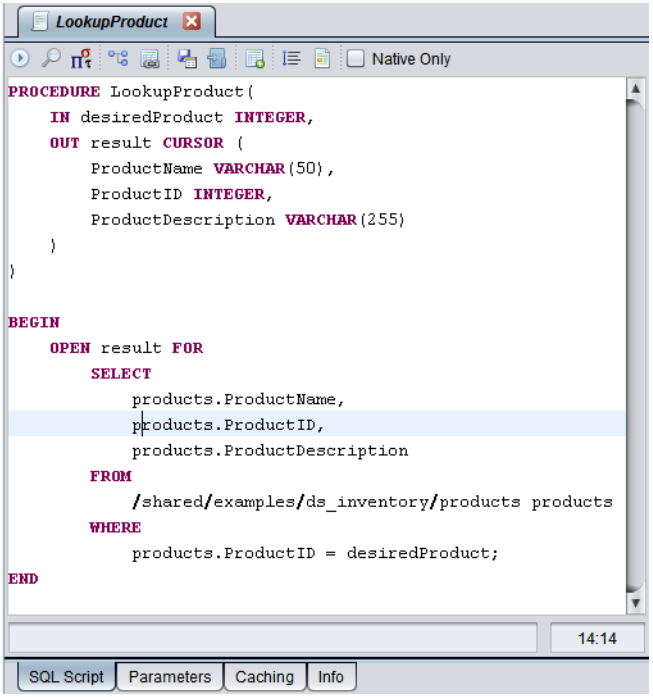
   — Keywords can be used in a SQL script, as long as you enclose them in double quotes; for example:

```
SELECT “begin” INTO ...
```

— If you want to do an INSERT, UPDATE, or DELETE operation, you must include the INSERT, UPDATE, or DELETE statement in a SQL script.

4. Save the script.

   Here is a sample SQL script. Note the error messages, and the row number and character position fields at the bottom of the panel (separated by a colon).



5. See these sections for more information:

   — For more information about working with the SQL script, see Working with SQL Scripts, page 270.

   — For information on how to define caching for your procedures, see TDV Caching, page 465.

   — For details on executing a SQL script, see Executing a Procedure or Parameterized Query, page 305.

Working with SQL Scripts

As part of creating a SQL script, you might also want to do the following:

Designing Parameters for a SQL Script

If you want to create a script by first designing its parameters, you must design the parameters in the Parameters Panel, page 667 and then write a procedure in the SQL Script panel.

Note: If a SQL script has an array as an input parameter, you can supply the input value when executing the script in Studio.

The Design Mode, which you use in this procedure, lets you formulate the input/output parameters for a procedure.

For more on SQL script execution, see Executing a Procedure or Parameterized Query, page 305.

To design the parameters for a SQL script in the Parameters panel

1. Create a new SQL script as described in Java Procedures, page 277.
2. Select the Parameters tab.
3. Check the Design Mode check box at the top right to make the Parameters panel editable.

   Studio displays a Design Mode Notes dialog box.

   The Design Mode check box lets you do a top-down design from the Parameters panel.

   When you use the Parameters panel to design the interface, the SQL implementation is not automatically updated in the SQL Script panel. You must make sure there is a match between the definitions in the SQL Script panel and the definitions in the Parameters panel using the design mode, including the order in which they are provided; otherwise, the script will generate an error when executed.
4. Review the warning and click OK.
5. Click the Add button to start adding parameters, and select a data type.

   When you click Add and select a data type, a new parameter is added with a default name and the specified data type.

Supported data types are:

- Binary—BINARY, BLOB, VARBINARY
- Decimal—DECIMAL, DOUBLE, FLOAT, NUMERIC
- Integer—BIGINT, BIT, INTEGER, SMALLINT, TINYINT
- String—CHAR, CLOB, LONGVARCHAR, VARCHAR
- Time—DATE, TIME, TIMESTAMP
- Complex—CURSOR, XML

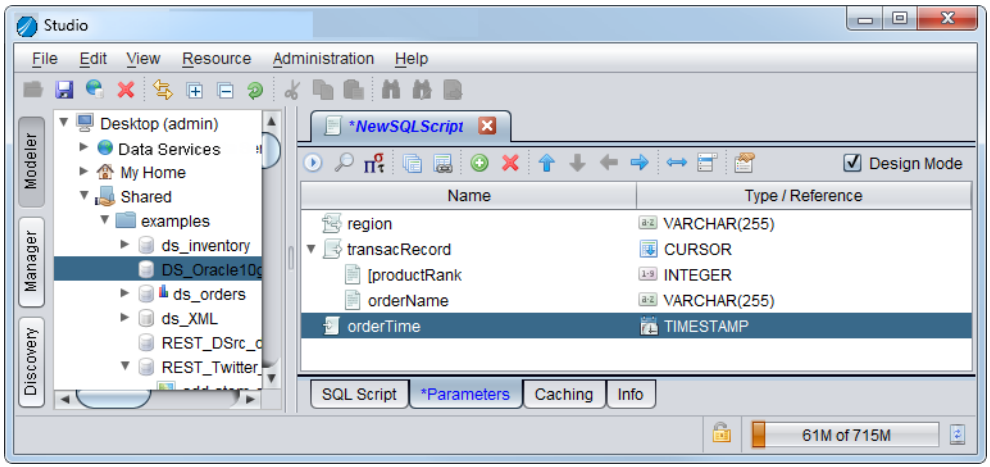
The Browse option is for choosing a definition set. See the topic Definition Sets, page 377, for details on definition sets.

6. Right-click the parameter name, choose Rename, and type the new name.
7. Optionally, change a parameter's data type:
   a. Right-click the data type name and select Change Type.
   b. Right-click the parameter name or data type, and select a data type from the drop-down list.
   c. Optionally, specify the length or the number of digits for certain data types.
   d. Click OK.
8. Select the parameter, and click Cycle I/O Direction to indicate whether the parameter is input, output, or both.

   The Cycle I/O Direction button is not available for parameters that are part of a cursor.
9. Use the navigation buttons to move a parameter up, down, left, or right.

   The up and down navigation buttons are enabled when there is more than one parameter.

The left navigation button is enabled for a CURSOR parameter. The right navigation button is enabled for a parameter in a row immediately after the last parameter in a CURSOR.



10. After specifying all of the parameters you want to define, save the edits.
11. In the SQL Script panel, complete the procedure using the parameters designed in the Parameters panel.
12. Save the script.

    After the parameters in the design match the parameters in the script on the SQL Script panel, the name of the script is displayed in black.

Promoting Procedures to Custom Functions

You can use a procedure with scalar output (that is, a procedure that returns a single value; it can have zero, one, or multiple inputs) to directly manipulate data as part of a SELECT statement, or as a criterion in a condition. (See Including a Function in a SELECT or WHERE Clause, page 234.) To use a procedure in this way, you promote it to a custom function.

Note: To *promote* a saved procedure is to add it to the drop-down list of custom functions in the Studio user interface, using the procedure described below.

An administrator can promote any SQL script or transformation using the Administration > Custom Functions menu option in Studio.

Note: A custom function cannot be rebound.

To promote a procedure to a custom function

1. Log in as the administrator, or have the administrator perform these steps for you.
2. Locate and open a SQL script. Make sure it has exactly one scalar output.
3. In Studio, use the Administration > Custom Functions menu.

   If the number of candidate SQL scripts is large, it might take Studio a while to prepare the list.
4. If you want to narrow the list, type a string in the Find field and click Search.

   To view the full list again after a search, clear the string from the Find field and click Search again.
5. Select the procedure to promote to a custom function.
6. Click OK.

This function will now be available for a view as a custom function.

Pushing a Custom Function to the Native Data Source

You can force a custom function to be executed by the native data source rather than by the TDV query engine. For a description of the types of SQL scripts that can be made into custom functions, see Promoting Procedures to Custom Functions, page 273.

To promote a SQL script to a custom function that is always pushed to a native data source, you use the Native Only check box, as described in the procedure below.

Note: The TDV query engine might push a custom function to the data source even without your checking the Native Only check box. The check box is available to guarantee that the custom function is pushed.

To push a custom function to the native data source

1. Log in as the administrator, or have the administrator perform these steps for you.
2. Locate and open a SQL script with one scalar output and check the Native Only check box.
3. In Studio, use the Administration > Custom Functions menu.

   If the number of candidate SQL scripts is large, it might take Studio a while to prepare the list.
4. Select the procedure to promote to a custom function.

5. Click OK.
6. Determine which adapter file you want to modify using the following guidelines and tips:
   - Changes made under /conf/adapters remain after applying hotfixes, and are included in any CAR file.
     This is true for /conf/adapters/custom and /conf/adapters/system.
   - Changes made to system adapters under /conf/adapters/system might require slightly more editing. For example, for Oracle you would need to define a <common:attribute> </common:attribute> for 11g.

   Note: If you want a custom function to be pushed to *all* versions of a data source, add the name and signature of that custom function to the "generic" capabilities file for that data source.

   - Changes should not be made under /apps/dlm.
   - You can choose to create a custom adapter to manage your customizations, but if you are adding missing capabilities it might not be recommended.
7. Locate the capabilities file to customize.

   For example, to add a custom function, ss10, intended for a custom Netezza data source, open the file:

```
<TDV_install_dir>\conf\adapters\system\<netezza_ver>\<netezza>.xml
```

8. From a text editor, open and add XML with the name, type, value, and configID of the custom function. For example, to add custom function ss10, add XML similar to this:

```
<common:attribute
xmlns:common="http://www.compositesw.com/services/system/util/comm
on">
 <common:name>/custom/ss10(~string,~string)</common:name>
 <common:type>STRING</common:type>
<common:value>ss10($1,$2)</common:value>
<common:configID>ss10(~string,~string)</common:configID>
</common:attribute>
```

9. Restart the TDV Server.

The custom function, when invoked, will be pushed to the native data source for execution. For example, when **`ss10`** is invoked in a SQL statement that includes, a Netezza table, the function will be pushed to the Netezza data source for execution.

Using Pipes and Cursors

Pipes and cursors can be used separately or in combination to solve a variety of challenges within TDV. They are useful when some preprocessing needs to be done to complex data—for example, stripping out rows where the condition is difficult to express in SQL, or returning a flattened row from an XSLT transformation on an XML JMS message.

Pipes

A pipe is a cursor that can have data inserted into it from many sources. TDV only sees pipes as OUT parameters for a procedure. You can use INSERT statements to get the data into the pipes for processing. For example:

```
PROCEDURE q(OUT name VARCHAR)
BEGIN
DECLARE c CURSOR (name VARCHAR);
CALL /shared/rtnPipe(c);
FETCH c INTO name;
END
```

Cursors

For TDV, a cursor is a result set returned from a single data source. A cursor can be thought of as a pointer to one row in a set of rows. The cursor can only reference one row at a time, but can move to other rows of the result set as needed.

Cursors can be used to perform complex logic on individual rows.

To use cursors in SQL procedures, you need to do the following:

- Declare a cursor that defines a result set.
- Open the cursor to establish the result set.
- Fetch the data into local variables as needed from the cursor, one row at a time.
- Close the cursor when done.

Static Cursor Example

```
DECLARE <cursor-name> CURSOR FOR <select>;
```

Dynamic Cursor Example

```
PROCEDURE p(OUT p_name VARCHAR)
BEGIN
DECLARE c CURSOR (name VARCHAR);
OPEN c FOR SELECT name FROM /shared/T;
FETCH c INTO p_name;
```

```
CLOSE c;
END
```

The dynamic cursor can be reopened:

- With the same query or a different query
- Only after being closed

```
PROCEDURE p(OUT p_name VARCHAR)
BEGIN
DECLARE c CURSOR (name VARCHAR);
OPEN c FOR SELECT name FROM /shared/T;
CLOSE c;

-- Reopen with same query
OPEN c;
CLOSE c;

-- Reopen with new query
OPEN c FOR SELECT name FROM /shared/U WHERE birthdate >
'2000-01-01';
CLOSE c;
END
```

Java Procedures

Java procedures are programs that you write in Java to access and use TDV resources. You can add these procedures as TDV Server data sources, execute them against a data source, use them in views or procedures, and publish them as TDV database tables or Web services.

For details about the TDV APIs for custom Java procedures and examples of Java procedures, see "Java APIs for Custom Procedures" and "Java for Custom Procedures Examples" in the *TDV Reference Guide*.

For details about adding a Java procedure to TDV Server, see Adding a Custom Java Procedure, page 167.

This topic contains:

For information on how to define caching for your procedures, see TDV Caching, page 465.

Viewing Java Procedure Parameters

When you add a Java procedure to the server, the procedure name is displayed in the Studio resource tree.

To view the parameters in a Java procedure

1. Open the Java procedure in the resource tree.

   The editor for the procedure opens on the right, displaying the Parameters panel.

   This panel displays the input or output parameters for the procedure. You can only *view* the parameters in the Parameters panel, because the parameters are defined and edited in the source Java code.

   Each parameter is displayed with its TDV JDBC data type and the data type native to the corresponding data source. The output parameters shown in this panel are rendered as columns in the result set when you execute the procedure.

For details on executing a Java procedure, see Executing a Procedure or Parameterized Query, page 305.

XQuery Procedures

XQuery procedures are programs that you write in XML to access and use TDV resources. You can add these procedures as TDV Server data sources, execute them against a data source, use them in views or procedures, and publish them as TDV database tables or Web services.

Note: The XQuery and XSLT procedures cannot be used as resources in the Any-Any transformation editor.

This topic contains:

For information on how to define caching for your procedures, see TDV Caching, page 465.

Creating an XQuery Procedure

This topic describes how to create an XQuery procedure.

To create an XQuery procedure

1. Select a folder in the Studio resource tree, and then right-click and select New XQuery Procedure, or select File > New > XQuery Procedure.
2. Supply a name for the procedure in the XQuery.

3. Type a name and click OK.

   The editor opens in the XQuery panel on the right.

4. Use the XQuery panel to formulate and edit the XQuery XML script:

   — You can use any non-declarative statement as the first statement after **BEGIN**.

   — If you have a script stored in your file system, you can upload it using the Insert from File button.

   — The XQuery panel can highlight XML syntax elements (comments, keywords, and so on).

   — Insert standard XML comments by highlighting one or more lines and typing:

   — Ctrl-Hyphen to insert two dashes (--) at the start of each line.

5. Save the script.

For details on executing an XQuery procedure, see Executing a Procedure or Parameterized Query, page 305.

Designing Parameters for an XQuery

If you want to create a script by first designing its parameters, you must design the parameters in the Parameters Panel, page 667 and then write a procedure in the XQuery panel.

For more on execution, see Executing a Procedure or Parameterized Query, page 305.

To design the parameters for an XQuery in the Parameters panel

1. Create a new XQuery procedure as described in XQuery Procedures, page 278.
2. Select the Parameters tab.
3. Check the Design Mode check box at the top right to make the Parameters panel editable.
4. Click the Add button to start adding parameters, and select a data type.

   When you click Add and select a data type, a new parameter is added with a default name and the specified data type.

Supported data types are:

- Binary—BINARY, BLOB, VARBINARY
- Decimal—DECIMAL, DOUBLE, FLOAT, NUMERIC
- Integer—BIGINT, BIT, INTEGER, SMALLINT, TINYINT
- String—CHAR, LONGVARCHAR, VARCHAR
- Time—DATE, TIME, TIMESTAMP
- Complex—CURSOR, XML

The Browse option is for choosing a definition set. See the topic Definition Sets, page 377, for details on definition sets.

5. Right-click the parameter name, choose Rename, and type the new name.
6. Optionally, change a parameter's data type:
   a. Right-click the data type name and select Change Type.
   b. Right-click the parameter name or data type, and select a data type from the drop-down list.
   c. Optionally, specify the length or the number of digits for certain data types.
   d. Click OK.
7. Select the parameter, and click Cycle I/O Direction to indicate whether the parameter is input, output, or both.

   The Cycle I/O Direction button is not available for parameters that are part of a cursor.
8. Use the navigation buttons to move a parameter up, down, left, or right.

   The up and down navigation buttons are enabled when there is more than one parameter.

   The left navigation button is enabled for a `CURSOR` parameter. The right navigation button is enabled for a parameter in a row immediately after the last parameter in a `CURSOR`. After specifying all of the parameters you want to define, save the edits.
9. In the XQuery panel, complete the procedure using the parameters designed in the Parameters panel.
10. Save the procedure.

After the parameters in the design match the parameters in the script on the XQuery panel, the name of the script is displayed in black.

XSLT Procedures

XSLT procedures are programs that you write in XML to modify XML documents—for example, to convert them to other formats based on an XSL style sheet. You can add these procedures as TDV Server data sources, execute them against a data source, use them in views or procedures, and publish them as TDV database tables or Web services.

The XSLT procedure editor is a stand-alone, language-specific processor that lets you compose and execute XSLT scripts.

Note: The XQuery and XSLT procedures cannot be used as resources in the Any-Any transformation editor.

This topic contains:

For information on how to define caching for your procedures, see TDV Caching, page 465.

Creating an XSLT Procedure

This topic describes how to create an XSLT procedure.

To create an XSLT procedure

1. Select a folder in the Studio resource tree, and then right-click and select New XSLT Procedure, or select File > New > XSLT Procedure.
2. Type a name for the procedure in the New XSLT Procedure dialog box and click OK.

   The editor opens in the XSLT panel on the right.
3. Use the XSLT panel to formulate and edit the XSLT XML script.
   - The XSLT procedure editor supports XSLT 1.0 and 2.0.
   - If you have a script stored in your file system, you can upload it using the Insert from File button.
   - The XSLT panel can highlight XSLT syntax (elements, attributes, and so on).
   - Insert standard XML comment notation by highlighting one or more lines and typing Ctrl-Hyphen.

The procedure editor inserts "<!--" at the start, and "-->" at the end, of each line you have highlighted.

4. Save the script.

For details on executing an XSLT procedure, see Executing a Procedure or Parameterized Query, page 305.

Designing Parameters for an XSLT Procedure

If you want to create a script by first designing its parameters, you must design the parameters in the Parameters Panel, page 667 and then write a procedure in the XSLT panel.

For more on execution, see Executing a Procedure or Parameterized Query, page 305.

To design the parameters for an XSLT in the Parameters panel

1. Create a new XSLT procedure as described in XSLT Procedures, page 281.
2. Select the Parameters tab.
3. Check the Design Mode check box at the top right to make the Parameters panel editable.
4. Click the Add button to start adding parameters, and select a data type.

   When you click Add and select a data type, a new parameter is added with a default name and the specified data type.

   Supported data types are:

   — Binary—BINARY, BLOB, VARBINARY

   — Decimal—DECIMAL, DOUBLE, FLOAT, NUMERIC

   — Integer—BIGINT, BIT, INTEGER, SMALLINT, TINYINT

   — String—CHAR, LONGVARCHAR, VARCHAR

   — Time—DATE, TIME, TIMESTAMP

   — Complex—CURSOR, XML

   The Browse option opens an Add Definition Type window in which you can select an XML schema definition set, select Type in the Show pane on the right, and select a type from the list. See Definition Sets, page 377 for details on definition sets.

5. Right-click the default parameter name, choose Rename, and type a new name.

6. Optionally, change a parameter's data type:
   a. Right-click the data type name and select Change Type.
   b. Right-click the parameter name or data type, and select a data type from the drop-down list.
   c. Optionally, specify the length or the number of digits for certain data types.
   d. Click OK.
7. Select the parameter, and click Cycle I/O Direction to indicate whether the parameter is input, output, or both.

   The Cycle I/O Direction button is not available for parameters that are part of a cursor.
8. Use the navigation buttons to move a parameter up, down, left, or right.

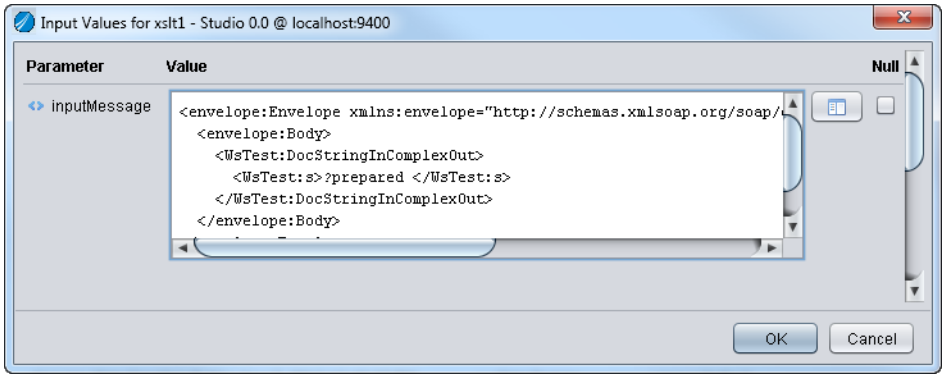
   The up and down navigation buttons are enabled when there is more than one parameter.

   The left navigation button is enabled for a CURSOR parameter. The right navigation button is enabled for a parameter in a row immediately after the last parameter in a CURSOR. After specifying all of the parameters you want to define, save the edits.
9. In the XSLT panel, complete the procedure using the parameters designed in the Parameters panel.
10. Save the procedure.

After the parameters in the design match the parameters in the script on the XSLT panel, the name of the script is displayed in black.

Parameter Entry

When you click the icon to execute an XSLT procedure with parameter input, an Input Values for <procedure_name> dialog box appears. It displays parameter names down the left side, and prepared XML code fragment with a question mark that you can replace with the input value.



You can replace the question mark with an input value, or check the Null box to eliminate the XML code fragment.

Packaged Queries

Packaged queries let you use database-specific queries defined within the TDV system and sent for execution on the targeted data source. Sometimes, you already have a complex query written for a particular database and it might not be feasible to rewrite that query in TDV SQL.

For example, your query might require a database-specific feature not available in TDV or perhaps the query takes advantage of the database-specific feature for performance reasons. In such cases, your database-specific query can be created as a packaged query and subsequently be used in other queries.

Multiple database-specific SQL statements can be executed sequentially by the same packaged query as long as only the last SQL statement returns a single cursor with at least a single column output. See the section on Multiple SQL Execution Statements in a Packaged Query, page 292.

Every packaged query is associated with a specific data source.

A packaged query is stored in TDV with the associated data source's metadata, and it functions like a stored procedure, accepting input parameters and producing rows of data. It must have exactly one output parameter that is a cursor with at least one column.

Because the TDV system cannot automatically determine the required inputs and outputs of a database-specific query, it is necessary for the user to supply this information.

For details on specifying parameters for a packaged query, see Specify Input Parameters for a Packaged Query, page 287.

For details on executing a packaged query, see Executing a Procedure or Parameterized Query, page 305.

This section includes:

Creating a Packaged Query

To create a packaged query

1. Right-click an appropriate location in the resource tree to add the packaged query, and select New Packaged Query.

   The New Package Query window opens.

2. Type a name for the packaged query.
3. Select the data source with which to associate the packaged query.

   If you want to associate the packaged query with a different data source later on, you can use the rebinding technique, which is described in the section Rebinding a Procedure, page 303.

4. Click OK.

   The editor opens on the right, displaying the SQL panel.

   Use the SQL panel to formulate and edit the SQL for the packaged query. The parameters you define in the SQL panel must match the parameters designed in the Parameters panel, including the order in which they are provided.

5. In the SQL panel for the packaged query:

   a. If the database-specific query already exists in the file system, upload the file using Insert from file in the editor's toolbar.

   The SQL from the file is copied onto the SQL panel.

   b. If the database-specific query is not available elsewhere, type it in the SQL panel. Add the string "<version 2>" to the beginning of the first SQL line

to handle null values passed by input parameters and to automatically escape any single-quote characters that might appear in the substitution values of any string or TIMESTAMP input parameters.

Packaged Query SQL does not require full pathnames to resources because the packaged query is already associated with the data source.

The packaged query must have exactly one output parameter that is a cursor with at least one column. In the following, the data source is orderdetails (specified in the FROM clause).

Define input parameters with curly braces, {N}, enclosing numerals that start from zero to N, as the input parameter placeholders (in this example specified in the WHERE clause). For details on defining input parameters and examples, see Specify Input Parameters for a Packaged Query, page 287.

Note: If the packaged query is equivalent to a single SELECT statement with a table or cursor output, check the Single Select check box. This flag tells the TDV query engine that the packaged query can be treated as a derived table, potentially enhancing the efficiency of the query plan. Because TDV does not process the packaged query, the user must make the determination whether the output qualifies for this optimization. Marking the packaged query as a Single Select allows TDV to add predicates (filter conditions) to the packaged query so that it can be a target of a semijoin optimization.

6. Click the Parameters tab to display the Parameters panel.

   Use the Parameters panel to design the input and output parameters for the query. The parameters defined in the Parameters panel must match the parameters defined in the SQL panel, including their order. After all the parameters are specified, you can use them to do a top-down design of the SQL in the SQL panel.

   To proceed with the example in Creating a Packaged Query, page 285, add one output parameter (UnitPrice) and two input parameters (ProductID and Status) in the Parameters panel.

   a. Select the output parameter named **result**, and click Add.

   b. Select Decimal > DECIMAL as the data type.

   A new parameter is added with the specified data type.

   c. Rename the newly added parameter as `UnitPrice` (to indicate the output), and click ENTER.

   d. Make sure that this parameter is a part of the CURSOR output by moving it to the right of result using the right-arrow button, if necessary.

   e. Add another parameter of the type Integer > INTEGER.

This is the first input parameter in our example. If it is placed under result as a part of the CURSOR output, use the left triangle button on the toolbar to move it outside the output cursor. A right-arrow icon appears next to it.

A right-arrow icon denotes an input parameter; a left-arrow icon denotes an output parameter.

f. Rename the parameter as ProductID, and click ENTER.

g. Add another input parameter named Status of the type String > CHAR.

7. Save the packaged query.
8. Execute the packaged query.

Executing a packaged query is similar to executing a parameterized query. For details, see Executing a Parameterized Query, page 306.

Specify Input Parameters for a Packaged Query

An input parameter is not required for a database-specific query. However, when creating a packaged query you can define one or more numbered input parameters to insert prior to execution and submission of the database-specific query. For each input value in the database-specific SQL, you must create an input parameter in the packaged query using the Parameters panel editor in Studio.

For example, your database-specific query might be used to fetch a row from a table but require an ID to identify the specific row. Packaged queries can easily define an ID input parameter or other parameter values for insertion at runtime.

When TDV Server executes a packaged query, it inserts actual values into the database-specific query before sending the packaged query to the data source for execution.

An input parameter consists of a name and a data type, both of which you specify when you define the parameter. The parameter name is displayed at query execution time, prompting for an input value. The parameter name has no effect on the database-specific query; it is used by users of your packaged query. However, the data type you specify does affect how a value is formatted when it is inserted into the database-specific query.

Data Types

Packaged queries support the following input data types:
`DECIMAL, DOUBLE, FLOAT, NUMERIC, BIGINT, BIT, INTEGER, SMALLINT, TINYINT, CHAR, LONGVARCHAR, VARCHAR, DATE, TIME, TIMESTAMP`

Input Substitution

The package query can define N inputs, with substitution patterns numbered from {0} to {N-1}.

Input Substitution Pattern

Packaged queries use one of two different input substitution patterns for each numbered input. The first line of the SQL must begin with <version 2>

- {N}—the numbered input is replaced with the data value with no changes.
- {N:string-sql-literal}—for data types of VARCHAR and TIMESTAMP the numbered input placeholder is replaced with the substituted string value and enclosed within single quotes.

  If any single-quote characters are present in the string value that is to be substituted, they are automatically escaped with a second single-quote.

  Null values are substituted.

Package queries that do not require an input, do not need a substitution pattern or substitution placeholders; however, the query must have exactly one output parameter that is a cursor with at least one column.

When **STRING** and **DATE** data types are used, you should use a single quote before and after the input substitution pattern. This rule does not apply to numeric data types.

Input Parameters Evaluation

This section illustrates how the packaged query input parameters are evaluated at runtime. The first six query examples are valid. The final two query examples are invalid.

Valid Query 1: Simple Substitution

In the following example `{0}` is a string, `{1}` is a number, and the package query is defined with the following:

```
<version 2>
SELECT customer.balance
FROM customer
WHERE customer.name = '{0}' AND customer.id = {1}
```

The pattern for a string is enclosed in single quotes. The first input value replaces all the occurrences of {0}, the second input value replaces {1}, and so on.

Valid Query 2: Multiple Substitutions

Each input substitution pattern is allowed to appear more than once in a packaged query:

```
<version 2>
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = {0} AND customer.zip = {2})
OR (customer.id = {1} AND customer.zip = {2})
```

Packaged Query before Substitution

```
<version 2>
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = {0} AND customer.zip = {2})
OR (customer.id = {1} AND customer.zip = {2})
```

Packaged Query after Substitution

Given actual input values like a first input value of 101, a second input value of 102, and a customer.zip value of 94403, the substitution would look like the following:

```
<version 2>
SELECT
customer.name, customer.balance
FROM
customer
WHERE
(customer.id = 101 AND customer.zip = 94403)
OR (customer.id = 102 AND customer.zip = 94403)
```

Valid Query 3: Escaping Characters to Stop Substitution

If a pattern such as '{i}' needs to be preserved, use a backslash (\) to escape the opening and closing curly braces ('\{i\}').

Packaged Query before Substitution

```
<version 2>
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = {0}
AND (customer.note = 'preserved \{1\}input')
OR (customer.id = {1} AND customer.zip = {2}
```

Packaged Query after Substitution

Substituting the same input values as the previous example:

```
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = 101
AND (customer.note = 'preserved {1} input')
OR (customer.id = 102 AND customer.zip = 94403
```

Note: The backslash (\) is removed from the original query. Escaping the second curly brace is optional.

Valid Query 4: Escaping Escaped Characters

If backslashes are part of a valid string (for example, '\{i\}') that should not be altered, add another backslash character ('\\{i\\}').

Packaged Query before Substitution

```
<version 2>
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = {0}
AND (customer.zip = '\\{2\\}')
OR (customer.id = {1} AND customer.zip = {2})
```

Packaged Query after Substitution

Substituting the same input values as the previous examples:

```
SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = 101
AND (customer.zip = '\{2\}')
OR (customer.id = 102 AND customer.zip = 94403)
```

Valid Query 5: {N:string-sql-literal} Option

If a query input is a string containing a single quote, the input placeholder should be {N:string-sql-literal} so that it is enclosed in single quotes and escaped by the packaged query to become two single quotes after substitution.

Packaged Query before Substitution

```
<version 2>
SELECT customer.balance
FROM customer
```

```
WHERE
customer.name = {0:string-sql-literal} AND customer.id = {1}
```

Packaged Query after Substitution

The first substitution value is “Michael’s son” and the second is “123”.

```
SELECT customer.balance
FROM customer
WHERE
customer.name = 'Michael''s son' AND customer.id = 123
```

Valid Query 6: Another Escape Example for Numeric Values

If a query input contains a number enclosed in curly braces such as‘{55}’, escape each curly brace with a backslash. There is no need to escape the curly braces, if the curly braces enclose a string such as ‘{the Second}’.

Packaged Query before Substitution

```
<version 2>
SELECT customer.balance
FROM customer
WHERE
customer.id = {0}
AND customer.name = ‘George {the Second}’
AND customer.id = ‘\{55\}’
```

Packaged Query after Substitution

```
SELECT customer.balance
FROM customer
WHERE
customer.id = 123
customer.name = ‘George {the Second}’
AND customer.id = ‘{55}’
```

Invalid Query A

The following query is invalid because substitution pattern input {1} is missing from the query.

```
SELECT customer.name, customer.balance
FROM customer
WHERE customer.id = {0} AND customer.status  = {2}
```

Invalid Query B

The following query is invalid because the database-specific query defines four inputs, but the packaged query has defined only three input substitutions. The fourth SQL input placeholder{3} makes the query invalid. You need to define one more input parameter. Also the customer.zip is an integer that does not need to be enclosed in single quotes.

```
SELECT customer.name, customer.balance
FROM customer
WHERE
customer.id = {0} AND customer.zip = '{1}'
AND customer.status = {2} AND customer.email = {3}
```

Multiple SQL Execution Statements in a Packaged Query

You can execute multiple SQL statements sequentially with a single packaged query. Multiple database-specific SQL statements can be invoked to create tables, modify tables, drop tables, insert rows, update rows, delete rows, invoke procedures, commit transactions, or roll them back. The sequence of SQL executions ends when an error occurs or after a SQL statement returns a result set.

Define a Multi-Part Packaged Query

Sequences of SQL statements can be sent to the data source through packaged queries as long as no SQL statement except the last one returns a result. The return of a result set cursor, or an exception, signals the end of the sequence of SQL executions.

The first line of SQL in the packaged query must set the multi-part separator character or set of characters. The following sample multi-part separator sets the SQL statement delimiter as two semicolons separated by a space:

```
<version 2> multipartseparator=;+;
```

This statement can be used to define other unique character sets to comply with any special data source query language requirements, but make sure the chosen delimiter is not used within any string, which can cause an unintended break.

Using a separator like ";" causes problems if a semicolon also shows up inside a string value substituted into the query.

Given definition of the multi-part separator, SQL statements are parsed and executed sequentially until the last statement returns a result set.

Note: If you execute anything that modifies the state of the connection in a way that survives beyond the current transaction, you can cause unexpected results for another user when and if the connection is put back into the pool.

Parameterized Queries

Studio can create SQL SELECT statements that include named parameters in the projections or selections. In the TDV system this feature is called a *parameterized query*. Such queries are implemented as single-statement SQL scripts. The SQL script code is automatically generated when you design a parameterized query with the Model and Grid panels. For details on SQL script languages, see the *TDV Reference Guide*.

The resources you can use in a parameterized query are tabular data and any procedure that outputs either a set of scalars or one cursor.

A parameterized query lets you include {param NAME TYPE} structures in the Grid panel to parameterize parts of the view. The resource itself is actually a SQL script with a model, so it functions like a procedure within the system.

The following sections describe how to create and execute a parameterized query:

Quick Steps for Creating a Parameterized Query

Here are some quick steps for using a table in a parameterized query. For details and diagrams, see Add Input Parameters, page 294.

To create a parameterized query

1. Select File > New > Parameterized Query from the menu bar, and give it a name.
2. Add the appropriate table to the Model panel on the right.

   See Adding a Resource to the Model Panel, page 221 for details.
3. Click the Grid tab.
4. On the Grid panel, specify a column to project, or leave the cell entry as * (to select all columns).
5. Right-click a blank Column cell, and select Parameter.

6. In the Add Parameter dialog:
   a. Type a unique name for the parameter in the Parameter Name field (for example, selectedValue).
   b. Select a data type from the drop-down list, and click OK.

   The entry is displayed in the Grid in the following format:

```
{param <parameter name> <parameter data type>}
```

   The corresponding Alias cell is automatically assigned a unique value (for example, Expr1).
7. Click the icon to execute the parameterized query.

   The Input Values for <resource> window opens and prompts you to provide a value for the parameter you added in a previous step.
8. Type in the value for the parameter and click OK to execute the query.

Add Input Parameters

You can define input parameters for the SELECT, WHERE, or FROM clause of a parameterized query.

- For the SELECT clause, add it from a Column cell in the Grid panel. See Adding a Parameter to a SELECT Clause, page 295.
- For the WHERE clause, add it from a Criteria cell in the Grid panel. See Adding a Parameter to a WHERE Clause, page 296.
- For the FROM clause, include a procedure with an input parameter in a parameterized query. See Adding a Parameter to a FROM Clause, page 296.

This section provides the details of adding parameters using an editor window (Add Parameter). You can also type the parameters in the Grid panel.

About Parameter Names

- When you type a parameter, it must be in the format {param <name> <type>}. The parameter name is case-sensitive.
- Several parameters can have the same name, provided they are distinguished by lowercase and uppercase letters, or by data type.
- Different parameters can have the same name and data type, if they have different aliases.

Adding a Parameter to a SELECT Clause

When you want parameters to be used in a SELECT clause, you must add them from cells in the Column column.

To include a parameter in the SELECT clause of a parameterized query

1. Right-click a location in the resource tree and select New Parameterized Query.
2. In the Input window, supply a unique name for the parameterized query, and click OK.

   The editor for building the query appears.
3. Add the resources to the Model panel.

   You can add any type of table (relational, LDAP, delimited file) and a procedure that contains scalar outputs or one cursor output.

   See Adding a Resource to the Model Panel, page 221 for details.
4. Click the Grid tab.
5. In the Grid panel, which works like the Grid panel in the view editor, click the first Column cell and select a column.
6. Right-click the next Column cell and select Parameter.
7. In the Add Parameter dialog, type a name in the Parameter Name field, and select a data type.

   This entry now appears in the Column cell.
8. Click the name in the Alias cell and rename it if you want.
9. Click the SQL Script tab if you want to view the auto-generated SQL script.
   - The parameterized query is treated as a PROCEDURE in the SQL script.
   - The parameter defined in a previous step appears as an input parameter of the procedure, and is also included in the SELECT clause of the parameterized query.
   - The output of the parameterized query is rendered as a CURSOR-type OUT parameter of the procedure.
10. Click the Parameters tab if you want to view how the parameters are displayed.
11. Save the parameterized query.

Adding a Parameter to a WHERE Clause

When you want parameters to be used in a WHERE clause, you must add them from cells in the Criteria column.

To include a parameter in the WHERE clause of a parameterized query

1. In the resource tree, right-click a container and select New Parameterized Query.
2. In the Input window, supply a unique name for the parameterized query and click OK.

   The editor for building the resource opens in the right pane.
3. Click the Model tab.
4. Drag and drop tables, views, or other procedures from the Studio resource tree into the Model panel.
5. In the Grid panel, click a Column cell to select one of the column names.
6. Click a second Column cell and select another column name.
7. Right-click the Criteria cell corresponding to the second column name under Column, and select Parameter.
8. In the Add Parameter window, supply a name for the parameter, select the data type, and click OK.

   This parameter appears in the Criteria column.
9. Click the SQL Script tab if you want to view the auto-generated SQL script.

   The parameter you added is placed in the parameterized query's WHERE clause.
10. Save the parameterized query.

Adding a Parameter to a FROM Clause

The following steps use a transformation as a sample procedure type to include in a parameterized query. The steps are similar for other types of procedures.

To include a parameter in the FROM clause of a parameterized query

1. Right-click a node in the resource tree and select New Parameterized Query.
2. In the Input window, supply a unique name for the parameterized query and click OK.

   The editor opens in the right pane.

3. Drag the transformation that requires input parameters and drop it onto the Model panel. For information on transformations, see Using Transformations, page 311.

   The Input Parameters for <transformation name> window opens.

   In this window, the upper section displays the input parameters. The icon next to each parameter represents its type.

4. To display the parameter's definition, click the Show Definition button.

   The window displays the parameter's name and type.

   In the **Value** group box, all options are mutually exclusive. Null specifies a NULL value; if it is disabled, the parameter cannot be NULL. Literal applies a value selected in the upper section. Query Parameter accepts a name for the parameter.

   Each parameter in the upper section is displayed with its initial value. If the parameter is nullable, the default is null. If the parameter is not nullable, the default value depends on type: zero for numeric, an empty string for a character, and the current date for a date.

5. Select the Query Parameter radio button, type the name, and click OK.

   Do not include string values in quotation marks. The parameter name must be unique and begin with a letter. It can contain any number of alphanumeric characters and can include an underscore. It must not contain any spaces.

6. Click OK to save the entry and close the window.

   The name of the parameter appears as the input parameter (IN) and is also included in the FROM clause.

7. If you want to view the auto-generated SQL script, click the SQL Script tab in Studio.

8. Save the parameterized query.

Here is an example of adding a parameter to the FROM clause of a parameterized query.

```
PROCEDURE paramQueryWithWebService(
    IN ticker VARCHAR(255),
    OUT result CURSOR (
        ID INTEGER,
        PARENT_ID INTEGER,
        "DEPTH" INTEGER,
        NAME VARCHAR(255),
        XPATH VARCHAR(255),
        "PATH" VARCHAR(255),
        "POSITION" INTEGER,
        "VALUE" VARCHAR(255)
        )
    )
    BEGIN
        OPEN result FOR
            SELECT
                *
            FROM
                /shared/sources/transformations/basic/trans_getQuote('
  <ns1:Body>
    <ns2:getQuote xmlns:soap-env="http://schemas.xmlsoap.org/soap/enve
      <symbol>' || ticker || '</symbol>
    </ns2:getQuote>
  </ns1:Body>
</ns1:Envelope>') trans_getQuote;
    END
```

The name of the parameter (ticker) appears as the input parameter (IN) and is also included in the FROM clause.

Physical Stored Procedures

TDV supports the introspection of data sources that contain stored procedures written in SQL. Procedures stored in data sources are referred to as *physical stored procedures*.

Note: In this document, the terms “physical stored procedure” and “stored procedure” are use interchangeably.

Physical stored procedures have certain capabilities and limitations:

- They can have scalar parameters, but the direction of scalar parameters might not always be detected.
- They can return one or more cursors that represent tabular output.
- Array-type parameters are not allowed in published stored procedures.

- If they are in a physical data source, you can manually specify cardinality for them. See Creating Cardinality Statistics on a Data Source, page 568.
- A stored procedure that returns only one cursor can be used within a view.
- A stored procedure that returns multiple cursors or scalar parameters cannot be included in a view.
- TDV supports the introspection of some stored procedures for some data sources.
- Cursors are not introspected for DB2, SQL Server, and Sybase databases.
- In Oracle data sources, the cursor type is introspected as type OTHER. If you do not change the cursor's signature but execute the Oracle stored procedure, cursor values are interpreted as binary data. However, if you edit the stored procedure and define the cursor's signature, its output is displayed in tabular form.

To edit stored procedures and define cursor signatures, see:

- Editing a Stored Procedure in an Introspected Data Source, page 299
- Editing a Stored Procedure in Microsoft SQL Server, page 302
- Editing a Stored Procedure in DB2, page 302

Editing a Stored Procedure in an Introspected Data Source

During introspection, the direction of the scalar parameters and cursors in stored procedures might be unidentified. A stored procedure with an unidentified scalar parameter or cursor direction returns an error when run as-is. In such cases you need to specify parameter and cursor direction manually.

This section uses an example to show how to specify scalar parameter or cursor direction, and define a cursor signature. The stored procedure discussed here has two scalar parameters (one input and one output) and it returns a single cursor that has seven columns.

To edit a stored procedure in an introspected data source, see:

- Specifying the Direction of Scalar Parameters and Cursors, page 299
- Defining a Cursor for Projection, page 300

Specifying the Direction of Scalar Parameters and Cursors

If introspection results include a parameterized query with scalar parameters whose direction is undefined, follow these steps to define the direction of those scalar parameters.

To specify the direction of a scalar parameter or cursor

1. Double-click the stored procedure to open it, or right-click it and select Open.

   The editor opens on the right.

   If the direction of an introspected parameter or cursor is unknown—whether it is IN, INOUT, or OUT—the icon next to it looks broken.

   If you create a copy of a stored procedure in Studio, Studio automatically introspects the procedure (that is, gathers metadata for it) and assigns the default I/O direction of IN to any parameter or cursor whose direction is unknown.
2. Select the Design Mode check box in the editor's toolbar so you can start editing the parameters.
3. Select the parameter whose direction is unknown, and use the direction arrows buttons on the toolbar to specify the direction.
4. Save the stored procedure.

Defining a Cursor for Projection

If introspection results include a parameterized query with one or more cursors whose direction is undefined, follow these steps (using the sample cursor and its parameters as a guide) to manually define a cursor for projection.

To define a cursor for projection

1. Double-click the stored procedure to open it, or right-click it and select Open.

   The editor opens on the right.
2. Select the Design Mode check box in the editor's toolbar so you can start editing the parameters.
3. Click the Add button, and select Complex > CURSOR.

   A new cursorParam of type CURSOR is added.
4. Select the cursorParam and click Cycle I/O Direction one or more times to change the direction to OUT.

   The other two valid direction icons are IN and INOUT.
5. With cursorParam still selected, click Add and select Time > TIMESTAMP.

   A new parameter named timestampColumn is automatically placed under cursorParam.
6. Rename the newly added parameter date and click Enter.

7. Add more child parameter names and types to cursorParam one at a time, in the following order, and specify each parameter's type as noted.

```
ord_num (type NUMERIC)
qty (type SMALLINT)
title_id (type CHAR)
discount (type DOUBLE)
price (type DECIMAL)
total (type DOUBLE)
```

   Your selection of the cursor columns is complete.

   If you are unsure what columns and data types to use for the cursor, use Studio's Design By Example tool to select the parameters. See Designing a Cursor by Example, page 301. When you use Design By Example, all previously defined parameters for the cursor are overwritten.

8. Select the parameter named RETURN_VALUE and click the Delete button, or select Delete from the right-click menu.

9. Save the stored procedure.

For details on execution, see Executing a Stored Procedure, page 307.

Designing a Cursor by Example

When you use Design By Example for a cursor, all previously defined parameters for the cursor are overwritten.

To design a cursor by example

1. Follow Step 1 through Step 5 as described in To define a cursor for projection, page 300.

2. Save the stored procedure after specifying the direction for any scalar parameter whose direction is unknown.

3. Click Design By Example on the procedure editor toolbar.

   The Input Values for <resource> window opens.

4. Supply a value for the scalar input parameter and click OK.

   The Design By Example window opens to display all the columns in the cursor, which is named result.

5. Click OK to accept the parameters.

   All the cursor columns are included for projection in the stored procedure.

   You can rename any of the parameters or columns, and also change their data types, as long as the data types are compatible with the ones in the underlying physical stored procedure.

6. Save the stored procedure.

For details on executing stored procedures, see Executing a Stored Procedure, page 307.

Editing a Stored Procedure in Microsoft SQL Server

The stored procedure discussed in this section is in a data source called Northwind in a Microsoft SQL Server database, with two scalar parameters whose directions are known. The stored procedure returns a cursor, which has two columns. The cursor will not be introspected during data source introspection. If you use the Design By Example window, both cursors will always be found in the same order (unless the stored procedure has changed because of introspection).

To edit a stored procedure that returns a single cursor

1. Open the stored procedure by double-clicking the stored procedure or right-clicking it and selecting Open.

   The editor opens on the right. This stored procedure has one input parameter and one output parameter.

2. You can design the cursor manually or by using the Design By Example tool, as described in Editing a Stored Procedure in an Introspected Data Source, page 299.

Editing a Stored Procedure in DB2

This section describes how to edit a stored procedure that does not have any scalar parameters but returns two cursors.

To edit a stored procedure that returns two cursors

1. Open the stored procedure by double-clicking the stored procedure, or by right-clicking it and selecting Open.

   The editor opens on the right. There is no scalar parameter in this stored procedure.

2. Check the Design Mode check box.

   This selection puts the editor in design mode, so you can start adding and editing parameters.

3. Add and edit parameters as required.

4. Click Design By Example on the toolbar, and use the scroll bar in the Design By Example window to view the two cursors.

   Use the expand/collapse symbol on the left of the parameter name to expand/collapse cursor display.

5. Click OK to include both the cursors for projection.
6. Save the stored procedure.

For details on executing stored procedures, see Executing a Stored Procedure, page 307.

Using Stored Procedures

Using stored procedures involves these activities:

Rebinding a Procedure

A procedure depends on one or more underlying sources, and the procedure is considered “bound” to those underlying sources.

The following procedure bindings are available:

- A SQL script can be bound to a table or a procedure.
- A transformation can be bound to hierarchical data (XML or WSDL).
- A packaged query can be bound to a data source.

Rebinding a procedure is useful:

- If after creating a procedure you decide to rebind the procedure to different sources.
- If an execution error occurs because a source no longer exists and you want to rebind the procedure with a new source.

If you modify a procedure or its dependencies, the binding is automatically updated when you save the script, and the Rebind panel reflects the change. For example, if your procedure has a data source dependency named ds_orders, and you rename it ds_orders_renamed and save the script, the Rebind panel lists the dependency as ds_orders_renamed.

To rebind a procedure to a different source

1. Open the procedure.
2. Click the Show Rebind Panel toolbar button.

   The Rebind panel opens in the lower section of the procedure editor, and displays the immediate dependencies as determined by the last-saved version of the procedure. (If a procedure has never been saved, the Rebind panel is blank.)
3. In the Rebind panel, highlight the source in the left pane and click Rebind.
4. In the Rebind dialog that opens, specify the source with which to rebind the script and click OK.
5. Save the procedure.

Setting Transaction Options for a Procedure

You can force Studio to ensure consistent execution results for a given set of input parameter values (variant). If the procedure is executed more than once for a variant, it returns the result set from the first execution. Results from these procedure variants are valid only for the lifetime of a transaction unless caching is enabled.

Note: When transaction procedure caching is enabled, results are cached until the transaction is completed. This can affect performance if the procedure results are unusually large or the transaction remains unresolved for a long time.

Studio also minimizes repeated calls to the data source—for example, if a procedure has an input parameter like this:

```
getAccountHistory(IN acctNum INTEGER, OUT results CURSOR)
```

If you call this procedure for the first time in a transaction with a given input value, the procedure runs as usual. If you call the same procedure again with the same input value in the same transaction, Studio returns the results of the previous call instead of re-executing the procedure.

To set transaction options

1. On the Info tab of the procedure definition, check the box labeled Execute only once per transaction for each unique set of input values to ensure consistent results during a single execution.

Executing a Procedure or Parameterized Query

You can initiate execution of a procedure in various ways:

- Executing a Procedure in Studio, page 305
- Executing a Parameterized Query, page 306
- Executing a Stored Procedure, page 307
- Executing a Procedure through JDBC, page 308

Executing a Procedure in Studio

When you execute a procedure in Studio, you have the option (with SQL scripts and parameterized queries) to use the **Trace** button to see line-by-line execution details. If input parameter values are required for the procedure, a window opens with the parameter names and fields for you to type the values before execution proceeds.

To execute a procedure in Studio

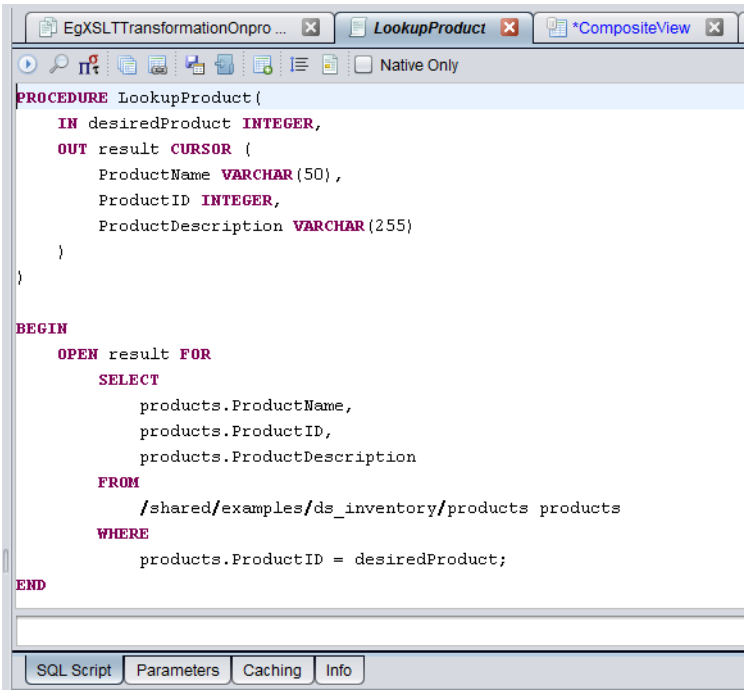
1. Open the procedure.
2. If you want to see line-by-line execution details, set the Trace button.
3. Click the Execute button.
4. If input parameter values are required, supply those values and click 0K.
5. Execute the procedure.
6. Review the execution details for each line are shown in the Console tab.
7. You can cause debug messages to be posted to the console or log by calling print, log, or logError.

Executing a Parameterized Query

This section provides examples of executing a parameterized query, an XSLT transformation, and physical stored procedures that return scalar parameters and cursors.

When you execute a parameterized query that has input parameters, you are prompted to supply values for them.

The parameterized query used here has one input parameter (desiredProduct) and an output cursor.



To execute a parameterized query

1. Double-click to open the parameterized query, and click the Execute button in the editor.

   The window opens so you can provide parameter values.

   This window displays the input parameters defined for the query. The icon to the left of a parameter represents the parameter type. To see the type as text, let the cursor hover over the icon. For XML parameters you might have access to a button on the window so that you can open up an XML editor.

2. Type a value in the Value field, or check Null (if the parameter is nullable).

For this example, supply a value for the defined input parameter, desiredProduct.

3. Click OK.

   When the parameterized query is executed, the product name, ID and description are displayed.

4. In the Result For result panel:

   a. To view the next fifty rows in a large data set, click Next.

   b. To view the details of a specific row in a Result Details dialog, select the row and click Details. To see details for an adjacent result, click Prev or Next. For some extremely large results, such as those generated by using EXTEND on an array to generate BIGINT number values, some of the data might be truncated without a visual indication by Studio.

   c. To save the results, use Save to File.

   d. To clear what is displayed in the tab, click Clear.

   Note: Result for result appears on the tab of the Result panel because result was the name used for the output parameter in the query. If you assign a different name to the output cursor, such as productFound, the tab would show Result for productFound.

Executing a Stored Procedure

This section describes how to use Studio to execute a stored procedure that returns one or more cursors. A stored procedure can contain any number of cursors.

To execute a stored procedure that returns a single cursor

1. Edit the stored procedure as described in Editing a Stored Procedure in an Introspected Data Source, page 299.

   The following steps apply to the stored procedure described in Designing a Cursor by Example, page 301.

2. Save the stored procedure, and click Execute.

   Because the stored procedure has a scalar input parameter, the Input Values for <resource> dialog opens. See Executing a Parameterized Query, page 306 for a description of this window.

3. Supply a value for the scalar parameter and click OK.

   The stored procedure is executed and the cursor output is displayed.

To execute a stored procedure that returns two cursors

1. Edit the stored procedure as described in Editing a Stored Procedure in an Introspected Data Source, page 299.
2. Save the stored procedure.
3. Click Execute.

   You can view the results of cursors in any order. However, due to JDBC driver constraints, all rows of the first-opened cursor need to be buffered before the rows of the next cursor are returned. Therefore, if you are interested in the results of the second cursor and want to improve performance, close the first cursor to disable buffering.

Executing a Procedure through JDBC

The name of the sample stored procedure is SP_D1. The username and password are guest and password. This section contains sample Java code to execute a stored procedure that returns two cursors.

To execute a stored procedure through JDBC

1. Publish the stored procedure as a TDV data service in the following directory:

```
Data Services/Databases/ds
```

2. Write and compile the Java code, and save the class file in a directory.

   See Sample Code Using MultipleCursors, page 308.

3. From the directory where you saved the class file, run the following command:

```
java -classpath <path_to_csjdbc.jar_file>/csjdbc.jar\;.
<MultipleCursors>
```

   MultipleCursors is the name of the class file, and csjdbc.jar is the name of the JAR file containing the class file.

Sample Code Using MultipleCursors

```
import java.sql.*;

public class MultipleCursors
{
  private static final String COMPOSITE_URL =

"jdbc:compositesw:dbapi@localhost:9401?domain=composite&dataSource
=ds";

  private static final String COMPOSITE_DRIVER =
```

```
    "cs.jdbc.driver.CompositeDriver";

  private static final String COMPOSITE_USER = "guest";
  private static final String COMPOSITE_PASSWORD = "password";

  public static void main(String[] args) {
    try {
      Class.forName(COMPOSITE_DRIVER);
    } catch (ClassNotFoundException ex) {
      ex.printStackTrace();
      return;
    }

    try {
      executeProcedure();
    } catch (SQLException ex) {
      ex.printStackTrace();
      return;
    }
  }

  private static void executeProcedure()
    throws SQLException
  {
    Connection conn = DriverManager.getConnection(
      COMPOSITE_URL, COMPOSITE_USER, COMPOSITE_PASSWORD);

CallableStatement stmt = conn.prepareCall("{call SP_D1(?,?)}");
    stmt.registerOutParameter(1, Types.OTHER);
    stmt.registerOutParameter(2, Types.OTHER);
    stmt.execute();

    printResultSet((ResultSet)stmt.getObject(1));
    printResultSet((ResultSet)stmt.getObject(2));

    stmt.close();
    conn.close();
  }

  private static void printResultSet(ResultSet rs)
    throws SQLException
  {
    ResultSetMetaData metaData = rs.getMetaData();
    int rowIndex = 0;
    while (rs.next()) {
      System.out.println("Row " + rowIndex++);
      for (int i=1; i<=metaData.getColumnCount(); i++) {
        System.out.println("  Column " + i + " " +
metaData.getColumnName(i) +
                           " " + rs.getString(i));
      }
    }
  }
}
```

Using Design Procedure By Example for Introspected Procedures

You can use Studio or you can use the command line to complete the steps. These steps help make use of database procedures that have cursor outputs that might not be detectable by TDV until after the procedure has been executed by TDV.

These steps are helpful if you have Oracle procedures with cursor outputs that you want to use in further data customizations within TDV.

To use the design procedures by example functionality

1. Execute the procedure. For more information, see Executing a Procedure or Parameterized Query, page 305.
2. Provide valid input values to the procedure.
3. Review the cursor output and review the data type information.

   TDV automatically does this for you, but reviewing at this stage can help you decide how you might want to make use of the data.
4. Save the cursor and data type information.
5. Use the saved information to design other TDV resources.

Alternatively, you can make use of the following API call from the command line or within a script:

```
designProcedureByExample <path> <inputs> <commitChanges>
```

- Provide the path to the procedure with the output cursors that you want TDV to execute.
- Provide valid input values for the procedure that you want TDV to execute.
- Set commitChanges to TRUE to save the output cursor metadata to the TDV repository for the procedure.

Using Transformations

This topic describes how to design transformations that are saved as procedures to query and manipulate data stored in a data source

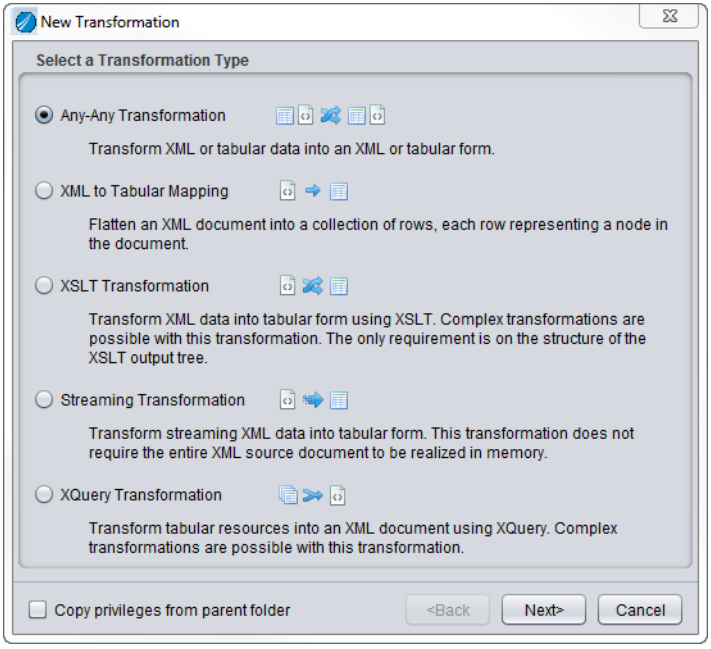
The data for modeling can come from a mix of relational (tabular) and hierarchical (such as XML) sources. Having tools to help you transform your data is helpful when designing your data virtualization layer.

This section assumes that you understand XML, XQuery, and WSDL.

Information about the Any-Any Transformation Editor is in:

Studio Transformation Types

Studio has a built-in transformation mechanism.



You can use to create the following types of transformations:

Type	Description
Any-Any Transformation	The Transformation Editor allows you to transform data from any type of data source structure to any type of target. The transformation of the data will happen in real time. Where possible the queries are pushed to the data source for execution. Data will be streamed, if possible, from the source to the target.
XML to Tabular Mapping	This transformation accepts an XML or WSDL source as input, and generates a tabular mapping to the elements in the source schema. The mapping can be thought of as producing a set of rows, each of which represents a node in the original XML document.

Type	Description

XSLT Transformation	The XSLT transformation lets you define how the XML data should be transformed using a graphical editor in Studio. It accepts an XML or WSDL source as input. Complex transformations are possible by writing custom XSLT. The only requirement is that the XSLT must produce tabular data with the following structure:
	`<results>`
	`  <result>`
	`  <column_one>a</column_one>`
	`  <column_two>b</column_two>`
	`  <column_three>c</column_three>`
	`  </result>`
	`  <result>`
	`  <column_one>d</column_one>`
	`  <column_two>e</column_two>`
	`  <column_three>f</column_three>`
	`  </result>`
	`</results>`

Streaming Transformation	The Streaming transformation lets you define how the XML data should be transformed using a graphical editor in Studio. This transformation is useful for transforming a large amount of XML data. This transformation does not require the entire XML source document to be realized in memory.
XQuery Transformation	The term *XQuery* stands for XML Query, a query that returns results in the form of an XML document. An *XQuery transformation* is a TDV resource that reads data from related sources, such as tables, views, and procedures, and assembles that data into an XML document that conforms to a user-specified schema called the *target schema*. An XQuery transformation looks and acts like any other procedure in the TDV system. Underlying an XQuery transformation is an *XQuery procedure* that is run by an XQuery processor. The target schema for the output XML document can be created or edited in the definition set editor. Studio also provides a graphical editor for mapping the target schema and tabular sources that are to supply the data for the XQuery in the transformation.

Creating an XML, XSLT, or Streaming Transformation

The steps you follow to create an XML to tabular, XSLT, or streaming transformation include:

- Adding a New Transformation (XML, XSLT, or Streaming), page 315
- Adding an XSLT Transformation for a Table, a View, or a Function, page 316
- Mapping Source Data to Target Output Columns (XSLT or Streaming), page 316 (XSLT and streaming only)
- Adding Target Columns through the Outputs Panel (XSLT), page 319 (XSLT only)

Adding target output columns is optional.

XSLT Transformation Limitation

The Data Map panel of the XSLT transformation does not display input fields for tables, views, and functions. Because the XSLT transform requires that the object being transformed has a Schema to generate Input Fields for the object.

TDV automatically generates a schema for XML-File data sources and WSDL data sources.

XML to Tabular Mapping

This transformation accepts an XML or WSDL source as input, and generates a tabular mapping to the elements in the source schema.

The mapping can be thought of as producing a set of rows, each of which represents a node in the original XML document. The XML document is converted into a table that represents the document's data and nodes, as described in the following table.

Column Name	Column Type	Description
id	INTEGER	Unique identifier for an element or attribute value.
parent_id	INTEGER	Unique identifier for the parent of this element.
depth	INTEGER	Number of elements in the path.
name	VARCHAR	Local name of the element or attribute.
xpath	VARCHAR	Xpath expression that fully describes the location of the element or attribute.

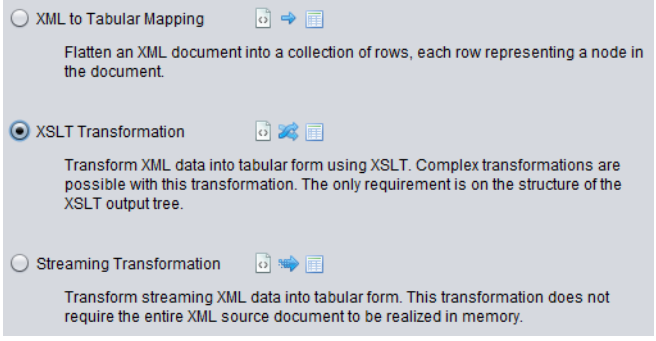
Column Name	Column Type	Description
path	VARCHAR	Path to the element or attribute. The path consists of a forward-slash-separated list of enclosing element names, ending with the local name of the element or attribute.
position	INTEGER	Position of this element relative to the path attribute.
value	VARCHAR	Value of the element or attribute.

Adding a New Transformation (XML, XSLT, or Streaming)

The steps to add an XML, XSLT, or streaming transformation are the same.

To create an XML, XSLT, or streaming transformation

1. Have an XML or WSDL source for the transformation ready, and know its location on the server.
2. From the resource tree, right-click and select **New Transformation**, or select File > New > Transformation.
3. In the New Transformation window, specify the type of transformation (XML to Tabular Mapping, XSLT Transformation, or Streaming Transformation), and click Next.



4. Locate and select an appropriate XML source for the transformation in the displayed tree.
5. Type a name for the transformation in the **Transformation Name** field.
6. Click **Finish**.

When you click **Finish**, the transformation is added to the resource tree, and the editor for the transformation opens in the right pane.

If the transformation is of the type XML to Tabular Mapping, it is ready to be used like any other procedure. In other types of transformations—**Streaming** and **XSLT**—the source elements and target output columns need to be mapped. (See Mapping Source Data to Target Output Columns (XSLT or Streaming), page 316.)

Adding an XSLT Transformation for a Table, a View, or a Function

The XSLT transform requires an XML schema and you can only use it indirectly on a table, view or function.

To add an XSLT transformation on a table, view, or function

1. Create an XML Definition Set to act as an XML schema if one does not exist.
2. Within Studio, create a procedure to act as a wrapper for the object.
3. Edit the procedure so that it uses a tag from the XML Definition Set as its return type.

Mapping Source Data to Target Output Columns (XSLT or Streaming)

The **Data Map** panel (see Data Map Panel, page 668) is available for XSLT transformations and streaming transformations. This panel lets you create and modify target output columns, map elements from the source XML document to the targets.

The following guidelines apply when using the **Data Map** panel:

- When you map source elements to target output columns in the **Data Map** panel, the XSLT is automatically generated in the **XSLT** panel.
- If you edit the text in the **XSLT** panel, the **Data Map** panel is permanently disabled. In this situation, design the outputs in the **Outputs** panel.
- You can map a source to more than one target, but you cannot map a single target to more than one source.
- The data types of the source and the target should match. Even if the data types do not match exactly, the link is valid if one data type is castable to the other.
- The target name is displayed as a column name in the output when the transformation is executed.
- When a link is selected, it is displayed as a thick line.

On the **Data Map** panel, you can:

- Manually create target columns and map them to source XML data.
- Automatically create target columns and map them to source XML data.
- Rename a target column and change its data type.
- Unlink or move targets.

To manually create target columns and map them to source XML data

1. If necessary, open the transformation.

   The transformation editor displays the **Data Map** panel.

2. If necessary, expand the nodes in the **Source** column to display all XML source definitions.
3. In the **Target** column, select the **outputs** node or an existing column, click **Add** ( ) in the toolbar, and select the data type for the output column from the drop-down list.

   An output column with a default name is created. You can create as many target columns as you need.

4. Connect a source and a target by selecting them individually and clicking the **Create Link** button.
5. Save your edits.

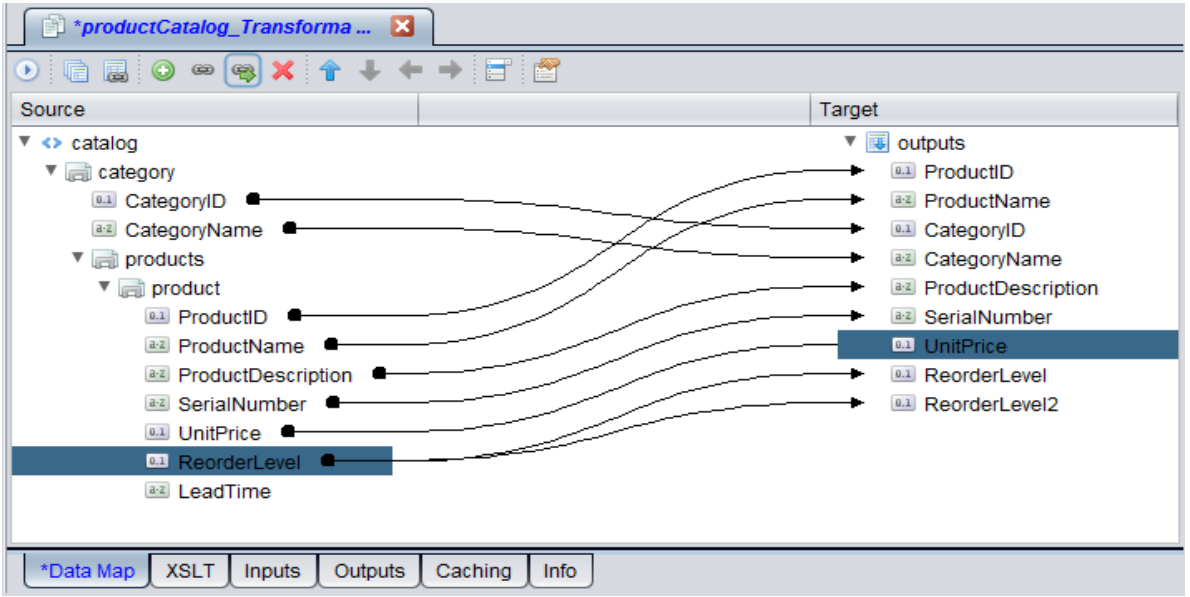
To automatically create target columns and map them to source XML data

1. If necessary, open the transformation.

   The transformation editor displays the **Data Map** panel.

2. If necessary, expand the nodes in the **Source** column to display all XML source definitions.
3. Select the source in the **Source** column.
4. Click the **Create Link And Target** button.

A target with the same name and data type of the source is created in the **Target** column, and a link is also created between the source and the target.



Selecting multiple sources and clicking **Create Link And Target** creates a separate target and link for each source. Selecting a source and clicking **Create Link And Target** several times creates that many targets and links pointing to the same source.

5. Save your edits.

To rename a target column and change its data type

1. If you want to rename a target, right-click the column name, select **Rename**, and supply a new name.
2. If you want to change the target's data type:
   a. Right-click the target name and select **Change Type** from the menu
   Or
   Select the target and click the **Change Type** toolbar button.
   b. In the **Choose Data Type** window, click the data type name and select a data type from the drop-down list.
   c. Optionally, specify the length or the number of digits for certain data types.
   d. Click **OK**.

To unlink or move targets

1. To unlink a target and a source, grab and drag the link away from the target or source, and release the mouse button.

   You can also select the link and click the **Delete** button to unlink a target and a source.

2. To align a target with its source, highlight the target and move it up or down using the up or down button**.**

3. Save your edits.

Adding Target Columns through the Outputs Panel (XSLT)

When you add target output columns through the **Outputs** panel for XSLT transformations, the XSLT in the XSLT panel is not automatically updated. You must manually update the XSLT.

For descriptions of the panels, seeXSLT Panel for XSLT Transformations, page 668 and Outputs Panel, page 669.

You can map an element in the source document to an XML-typed column in the result. When you do this, the element's entire sub-tree (attributes, namespaces, child elements, and contents) in the XML source document becomes the text of the XML typed column. This includes the element itself.

This feature allows you to map a subtree of the source XML document to a column of type XML or VARCHAR in the transformation output cursor.

You cannot make such a mapping through the **Data Map** panel. You must make this mapping by customizing the XSLT that is generated.

To add target output columns through the XSLT and Outputs panels

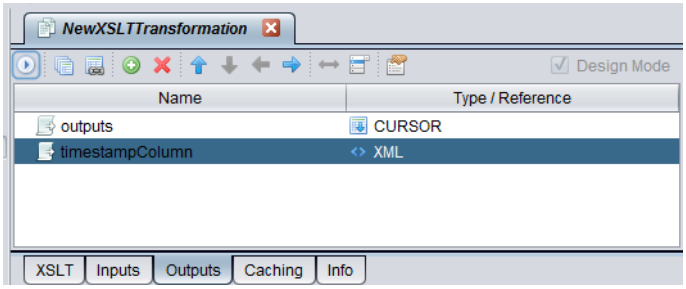
1. If necessary, open the transformation.

2. Edit the XSLT in the **XSLT** panel, as needed.

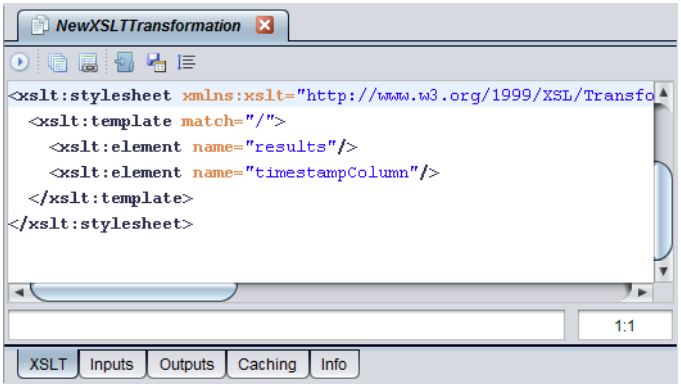
   If you edit the XSLT in the **XSLT** panel, the **Data Map** panel is permanently disabled, but the **Outputs** panel becomes editable (**Change Type**, **Delete** and **Rename** on a context menu) and the **Add** button in the **Outputs** panel toolbar is enabled.

3. Optionally, on the **Outputs** panel add target output columns, the same way you would design parameters for a SQL script on the **Parameters** panel.

See Designing Parameters for a SQL Script, page 271.



4. Update the XSLT in the **XSLT** panel to correspond to any output elements you added.



5. Save the transformation.

Creating an XQuery Transformation

This topic describes how to create an XQuery transformation.

To create an XQuery transformation

1. Have an XML definition set ready. The definition set provides the target schema for the output document.

   For details on creating a definition set, see Definition Sets, page 377.

2. Select a folder in the Studio resource tree, and then right-click and select New Transformation, or select File > New > Transformation.
3. In the New Transformation window, select XQuery Transformation and click Next.

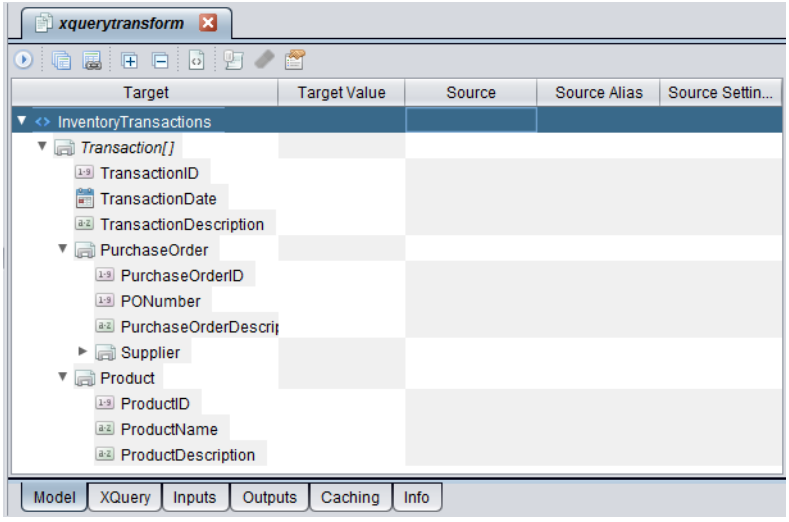
4. Supply a name for the transformation in the **Transformation Name** field.
5. Locate and select the XML definition set in the resource tree displayed in the left panel.
6. In the right panel, where all the elements in the XML definition set are displayed:
   a. (Optional) Expand the nodes and view the elements to see what is available.
   b. Select a top-level element.

   This top-level element determines the type and name of the root element in the output XML document.

   To have many top-level elements available for selection, use an XML definition set similar to UserSchema, which you can find in Data Services/Web Services/system/admin/
7. Click **Finish**.

   The transformation is added to the resource tree, and its editor opens in the right pane displaying the **Model** panel by default. The **Model** panel in the XQuery transformation lets you map source data to the target schema and transform tabular data into XML data.

   The following sample entries in the Model panel of an XQuery transformation named InventoryTransactions. Cells that are not appropriate to the transformation are dimmed; the other cells remain white.



8. Use the following sections to complete the definition of this transformation:
   - Specifying a Target Value, page 322
   - Viewing the Source Scope, page 323
   - Specifying the Source for a Top-Level Element, page 325
   - Specifying Settings for Target Sources in an XQuery Transformation, page 325
   - Specifying Global Input Parameters in an XQuery Transformation, page 327
   - Viewing and Editing the XQuery, page 327
   - Viewing Output Parameters in the XQuery Transformation, page 328

Specifying a Target Value

You can specify a value for any target that has a white cell in the Target Value column for its row. The Target column displays the XML schema that is uploaded (through the definition set) for this transformation. This is the target schema for the XML document that the XQuery would return. Expand the nodes in this column to view the target elements.

The tree-structure display in the Target column represents the hierarchical structure of the elements in the target schema. Elements followed by square brackets are unbounded, meaning that their occurrence is unlimited. Elements rendered in italicized type are optional.

The Target Value column (available for table columns, not for tables) contains the actual expressions (projections) that determine the values for the elements in the **Target** column. The expression (projection) in each cell is relative to the views, tables, and procedures that are currently in scope. The **Target Value** column is similar in functionality to the column named **Column** in the view editor's **Grid** panel.

To specify a target value

1. If necessary, click the **Expand All Rows** toolbar button to see the tables and columns you need to edit.
2. Click the cell below **Target Value** corresponding to the target table's row.
3. Select a value from the drop-down list, or type a value.
4. If the source settings are not displayed in the lower section of the editor, click the **Show Source Settings** toolbar icon, or click an icon in the **Source Settings** column.

Tabs open in the lower part of the editor showing the settings for a source: a join XPath expression (**Join** tab), a filter XPath expression (**Filter** tab), a sort order (**Sort Order** tab), or an input parameter (**Inputs** tab) as in a procedure. The **Schema tab** shows the schema for the selected source.

5. Specify a join relation between the current source and one of its direct ancestors in the **Join** panel.

   For example, **PurchaseOrderID = $INV/PurchaseOrderID** means that **PurchaseOrderID** in **PurchaseOrder** is joined to `PurchaseOrderID` in the ancestor **InventoryTransactions**.

6. Specify a filter on the current source (the equivalent of a `WHERE` clause in a `SELECT` statement) in the **Filter** panel.

   For example, **$INV/TransactionID > 25** is a filter on **TransactionID** in the **Source** named **inventorytransactions**. The filter is used to constrain the data returned from this resource.

7. Specify the order in which the results from the current resource are to be sorted for the XML document (the equivalent of an `ORDER BY` clause in a `SELECT` statement) in the **Sort Order** panel.

8. Specify the values for a source's inputs if the source is a procedure with input parameters in the **Inputs** panel.

9. Click the icon in the **Source Settings** column at any time to view or edit the setting.

Viewing the Source Scope

Before you view drop-down lists in the **Target Value** column, you might want to widen the column so you can see entire names.

The **Source** column (available for tables) lets you specify the tabular sources that provide data for the resulting XML document. Each source specified in this column corresponds to a top-level element (non-leaf node) in the target schema.

Resources specified in the **Source** column exist in a "scope" relative to the target XML document to which they provide data. At a particular location in the document, the source scope is defined as:

- The current resource (table, view, or procedure)
- All the resources that are directly above this resource in the document (direct ancestors)
- The input to the XQuery

The scope defines what resources are available to the value expression.

The scope for **Transaction** is its own source, the `inventorytransactions` table, and the global input **SupplierName**.

The scope for **PurchaseOrder** is its own source, the `purchaseorders` table, its direct ancestor, the `inventorytransactions` table, and the global input **SupplierName** (line #2).

The scope for Supplier is its own source, the suppliers table, its direct ancestors, the purchaseorders and inventorytransactions tables, and the global input SupplierName (line #3).

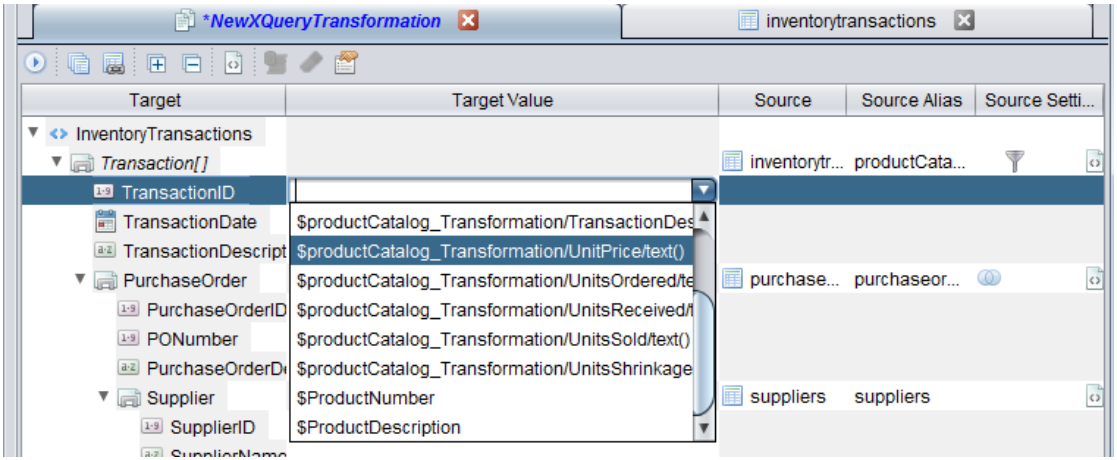
The scope for Product is its own source the LookupProduct procedure, its direct ancestor inventorytransactions table, and the global input SupplierName (line #4). The sources (purchaseorders and suppliers) in the scope of its sibling or peer resource (PurchaseOrder) are excluded from the scope of Product.

The Source Alias column (available for tables) lets you specify aliases for the sources in the Source column. The aliases are used when join conditions, filters, and inputs are supplied for a source value. This column is populated automatically when a resource is added to the Source column, but it can also be edited as a text field.

The Source Settings column (available for tables) displays icons indicating that certain settings have been specified for the corresponding source. When clicked, these icons display the corresponding settings in the lower section of the editor. For example, you can click the Schema icon to display the schema for the corresponding source.

To view the source scope for an element

1. Click a cell in the **Target Value** column.
2. View the drop-down list.



Multiple input parameters are shown at the end or the drop-down list, in the order in which they were defined in the **Inputs** panel.

Specifying the Source for a Top-Level Element

Top-level elements (which have leaf nodes under them) have white cells in the Source column corresponding to their rows. The Source column (available for tables) lets you specify the tabular sources that provide data for the resulting XML document. Each source specified in this column corresponds to a top-level element (non-leaf node) in the target schema.

To specify the source for a top-level element

1. Double-click the cell corresponding to a top-level element in the **Target** column; or select its top-level element in the **Target** column.
2. Click the **Choose Source** toolbar button).
3. In the **Choose a Table or Procedure** window that opens, select the source.

Specifying Settings for Target Sources in an XQuery Transformation

To specify the sources, values, and other settings for target sources in the

XQuery transformation

1. In the **Model** panel, specify the sources, target values, and source aliases to provide the data and constraints for the output XML document.

   In the current example, the top-level elements in the **Target** column are **Inventory Transactions**, **Transaction**, **PurchaseOrder**, **Supplier**, and **Product**. You can specify a source for each of these elements.

2. To specify a source:

   — Double-click the **Source** cell corresponding to the top-level element in the **Target** column.

   Or

   — Select the top-level element in the **Target** column and click the **Choose Source** toolbar button.

3. In the window that opens, select the source table and click **OK**.

4. If you want to change the automatically assigned alias, type a different alias in the **Source Alias** column for the source you just added.

   The aliases are used when join conditions, filters, and inputs are supplied for a source value.

5. To specify a value for a target, click the **Target Value** cell corresponding to the target in the **Target** column, and select a value from the drop-down list, as follows.

   For a target value, you can supply a literal value or a system function to be evaluated at runtime. Literal values must be enclosed in single or double quotes.

6. To specify a source setting, click the appropriate icon in the **Source Settings** column, or click the **Show Source Settings** toolbar icon to display the source settings panel in the lower section if it is not already visible. Then open the **Join**, **Filter**, **Sort Order**, or **Inputs** panel and supply the settings.

   `PurchaseOrderID = $INV/PurchaseOrderID` (in the **Join** panel) means that **PurchaseOrderID** in **PurchaseOrder** is joined to **PurchaseOrderID** in the ancestor **InventoryTransactions**. The syntax for the value expression `$INV/PurchaseOrderID` contains a reference to the alias (`INV`) for the parent source **inventorytransactions** preceded by the dollar sign (`$`).

7. To supply a value for an input parameter, if the source is a procedure that has input parameters, use the **Inputs** panel (in the source settings panel).

   a. Click the **Inputs** tab to open the **Inputs** panel.

   b. Click the **Inputs** icon in the **Source Settings** column.

In the **Inputs** panel, the names of the inputs are displayed, and the **Is NULL** check box is selected by default.

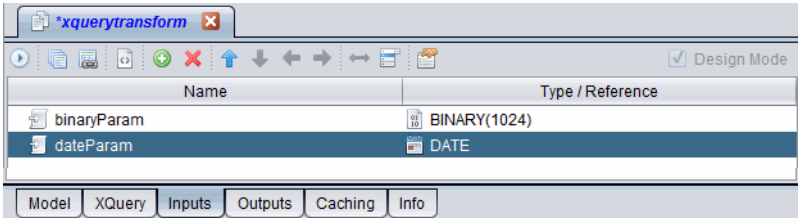
c. Click the row in the **Value** column, and select an appropriate value for the input parameter in the drop-down list.

Specifying Global Input Parameters in an XQuery Transformation

You can specify global inputs to the XQuery, which are subsequently available to be mapped (as values) to any compatible element in the target schema.

To specify global input parameters in an XQuery transformation

1. Click the **Inputs** tab (in the upper section of the editor).
2. Click **Add** in the toolbar, and specify the data type for the input parameter.



3. Rename the parameter as needed.

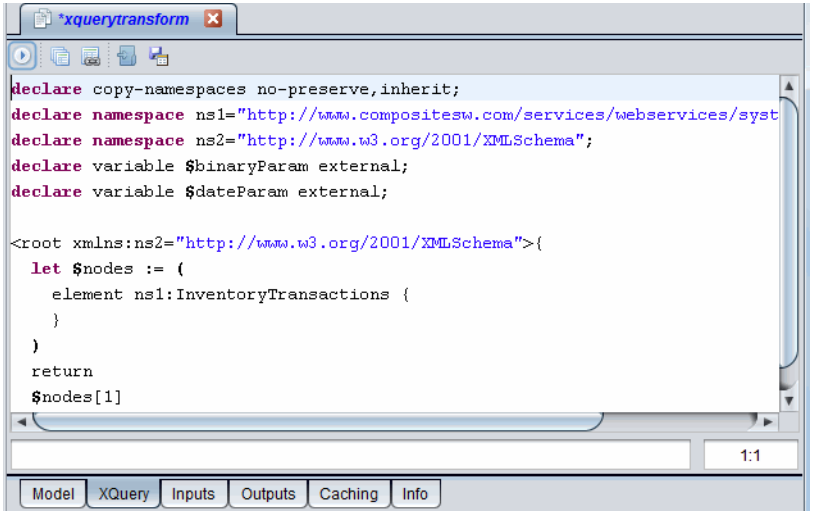
   This input parameter is available in the **Model** panel, and you can specify it as a target value for an appropriate target element.
4. (Optional) Click the **Outputs** tab to view the outputs.
5. Save the transformation.

Viewing and Editing the XQuery

After specifying all required sources, values, and other settings for target sources, you can view and edit the resulting XQuery.

To view and edit the XQuery

1. Click the **XQuery** tab in the editor to view the auto-generated XQuery for the model you designed in the **Model** panel.



2. View and edit the XQuery text in this panel as necessary.

   Editing the XQuery permanently disables the **Model** panel for the current transformation.

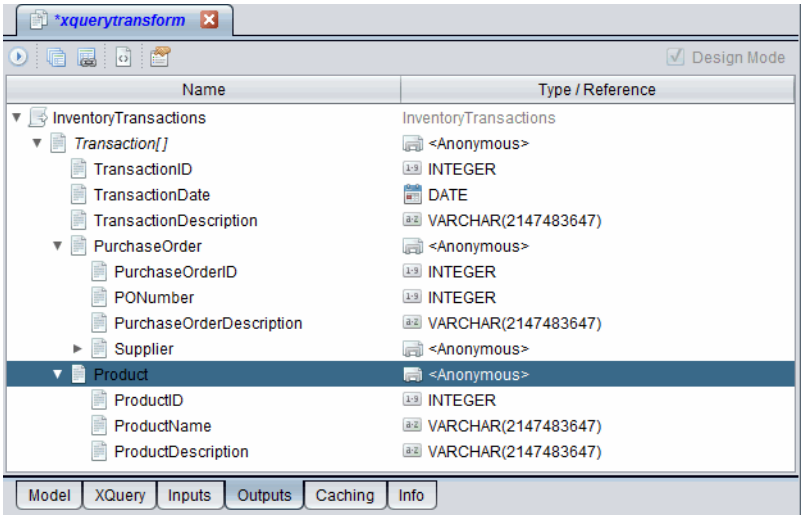
3. Save any changes you have made to the transformation.

Viewing Output Parameters in the XQuery Transformation

You can view the output parameters in the XQuery transformation.

To view the output parameters in the XQuery transformation

1. Specify the sources, values, and other settings for target sources as needed.
2. Select the **Outputs** tab in the editor to view the output parameters. The output parameters shown in the **Outputs** panel are rendered as elements in the output XML document.



For details on executing the transformation, see Executing a Transformation, page 331.

Upgrading XML to Tabular Mapping Transforms

If you have existing XML to Tabular Mapping transforms that you would like to update to be compatible and editable using the Any-Any Transformation Editor, you can follow the steps in this section to accomplish that. There are two different upgrade paths:

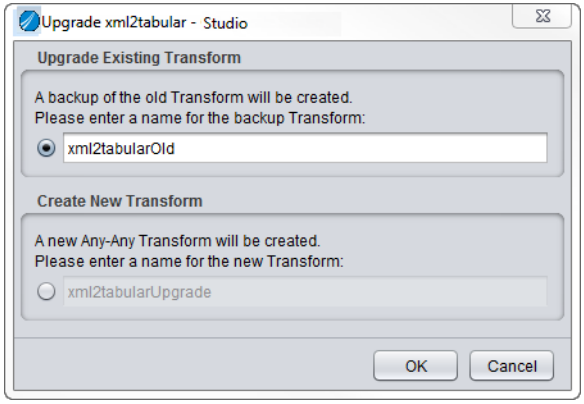
- Upgrading and Creating a Backup of an XML to Tabular Mapping Transform, page 330
- Creating an Upgraded XML to Tabular Mapping Transform, page 330

Upgrading and Creating a Backup of an XML to Tabular Mapping Transform

The steps in this task will keep the name of your existing transform as is and convert it to the new Any-Any style of transform, it will also create a backup XML to Tabular Mapping transform saved with the name given.

To upgrade and create a backup of an XML to Tabular Mapping transform

1. Make sure that your TDV Studio environment is running the latest service pack and that **`server_util.sh -reset namespace`** has been run to update the TDV Server and repository.
2. Open Studio.
3. Locate an existing XML to Tabular Mapping transform in your Studio navigation tree.
4. Select the transform, right-click and select Upgrade Transform.



5. Select the top radio button, accept or rename the transform to a name that will be easy to understand later.
6. Click **OK**.

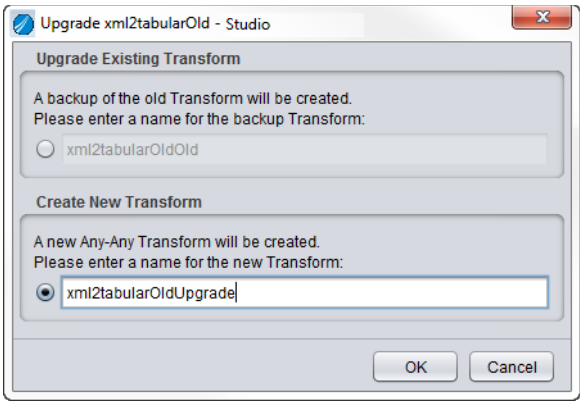
   The Studio navigation tree is updated. Your transform is converted and a new backup of the old XML to Tabular Mapping transform is add to the tree.

Creating an Upgraded XML to Tabular Mapping Transform

The steps in this task will rename your existing transform and convert it to the new Any-Any style of transform.

To upgrade an XML to Tabular Mapping transform

1. Make sure that your TDV Studio environment is running the latest service pack and that server_util.sh -reset namespace has been run to update the TDV Server and repository.
2. Open Studio.
3. Locate an existing XML to Tabular Mapping transform in your Studio navigation tree.
4. Select the transform, right-click and select Upgrade Transform.



5. Select the bottom radio button, and accept or rename the transform to a name that will be easy to understand later.
6. Click **OK**.
7. Click Refresh All to refresh the Studio navigation tree with the upgraded transform.
8. Optionally, open the new transform and edit it for your current needs.

Executing a Transformation

If the executed transformation is an XQuery transformation, you can save the result as an XML file and view it in a browser.

To execute a transformation

1. In Studio, open the transformation.
2. Click the **Execute** toolbar button.

If there is no need for an input parameter value, the procedure is executed immediately.

3. If you need to supply a value for one or more input parameters, type a value or select **Null** (default) for each field in the **Input Values for <resource>** dialog.

   Do not enclose string values in quotes.

4. In the **Input Values for <resource>** dialog, click **OK**.
5. If the executing task is successful, the **Result** tab displays the result set, fifty rows at a time.
6. In the **Result** panel:
   a. To view the next fifty rows in a large result set, click **Next**.
   b. To view the details of a specific row, select the row and click **Details**. For some extremely large results, such as those generated by using EXTEND on an array to generate BIGINT number values, some of the data might be truncated without a visual indication by Studio.
   c. To save the results, use **Save to File**.
   d. To clear what is displayed in the tab, click **Clear**.

Converting XML to JSON using Design By Example

If using one of the standard transform methods doesn't work for you for some reason, it is possible that you can convert XML to JSON using the Design By Example functionality that exists within TDV.

To convert JSON to XML form (or Tabular form)

1. Create a REST data source.
2. Select JSON format.
3. Add a new operation.
4. Introspect the operation.
5. Open the Operation and for Header/Body Parameters, select Design by Example.
6. Select an element.
7. Execute the operation.
8. Review the XML that is returned.

JSON XML List Formatting Example

Formatting lists using JSON can be done in many ways. Typically, they are done using a nested element in XML. The following example gives one option. There are many others. Consulting a good JSON reference can help you determine how best to format the list that you need to create.

```
"https://www.facebook.com/sacramentosports?ref=br_tf": {
"about": "Welcome to the official Sacramento Sport Facebook Page!",
"affiliation": "National Basketball Association, Western
Conference, Pacific Division",
"category": "Professional sports team",
"category_list": [
{ "id": "109976259083543", "name": "Sports Venue & Stadium" }

,
{ "id": "189018581118681", "name": "Sports Club" }

],
"checkins": 7149,
"is_published": true,
"location":
{ "street": "One Sports Way", "city": "Sacramento", "state": "CA",
"country": "United States", "zip": "95834", "latitude":
38.64906488403, "longitude": -121.51807577132 }
```

Using the Any-Any Transformation Editor

Before using the Transformation Editor there are a few concepts that you can familiarize yourself with. This topic also covers the basic usage of the Transformation Editor.

- Transformation Editor Concepts, page 335
- Using the Transformation Editor, page 340
- Caching Transform Data, page 375

Tutorials that walk you through the creation of several different types of Any-Any transforms can be found in the *TDV Tutorials Guide.*

Transformation Editor Concepts

The Transformation Editor is an editor within TDV that can be used to define the transformation rules for the source data sets. It can transform data from any source structure to any target structure. The transformation of the data happens in real time as data is streamed, if possible, from the source to the target. The source and target are not persistent structures.

A transformation is a procedure in TDV and it can be used by other resources in TDV as a procedure. When caching is enabled on a transform, it is cached in the same way as any other procedure in TDV. It produces XQuery code that is used to execute the transformation. The XQuery code is not directly editable.

The Transformation Editor can map and transform data to and from the following:

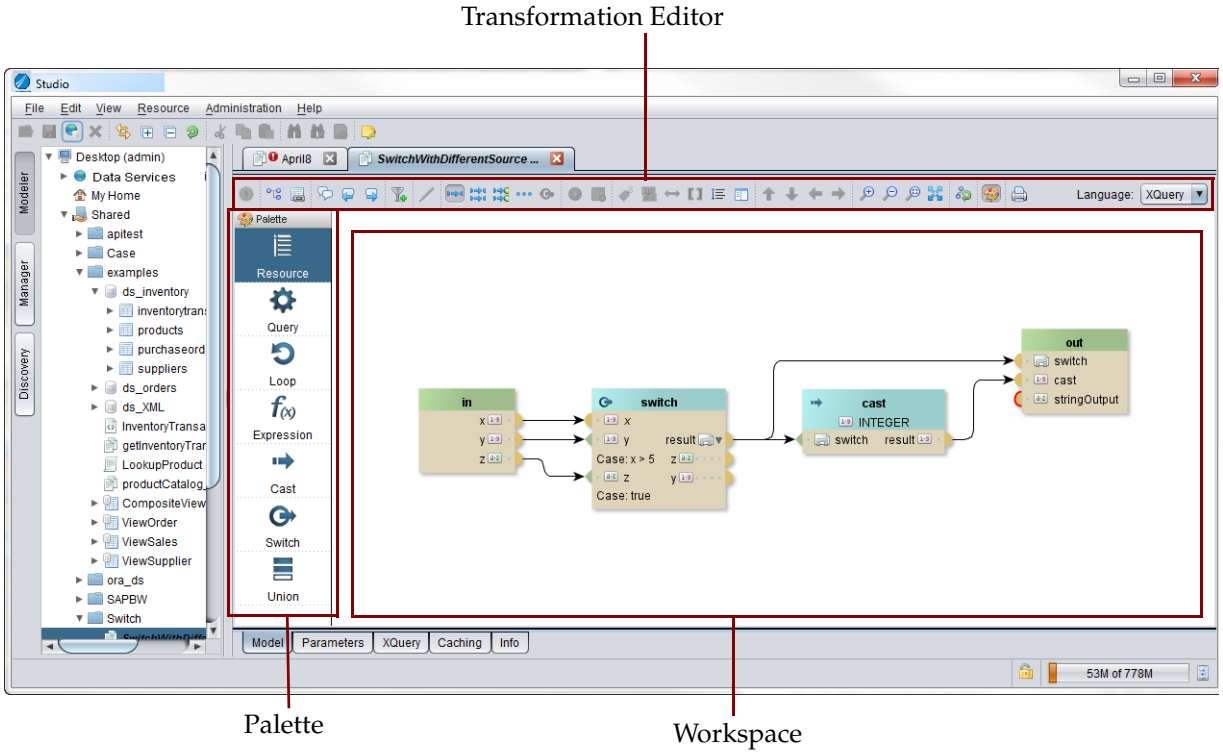
- Relational (Cursors)
- XML (Hierarchies)
- Scalar (Procedures)

You can also browse to and drag existing resources defined in Studio into the Transformation Editor. Concepts in other sections under this topic include:

- The Transformation Editor Window, page 336
- Transformation Editor Terminology, page 337
- About Transformation Editor Operations and Parameter Containers, page 337
- Transformation Editor Limitations, page 338

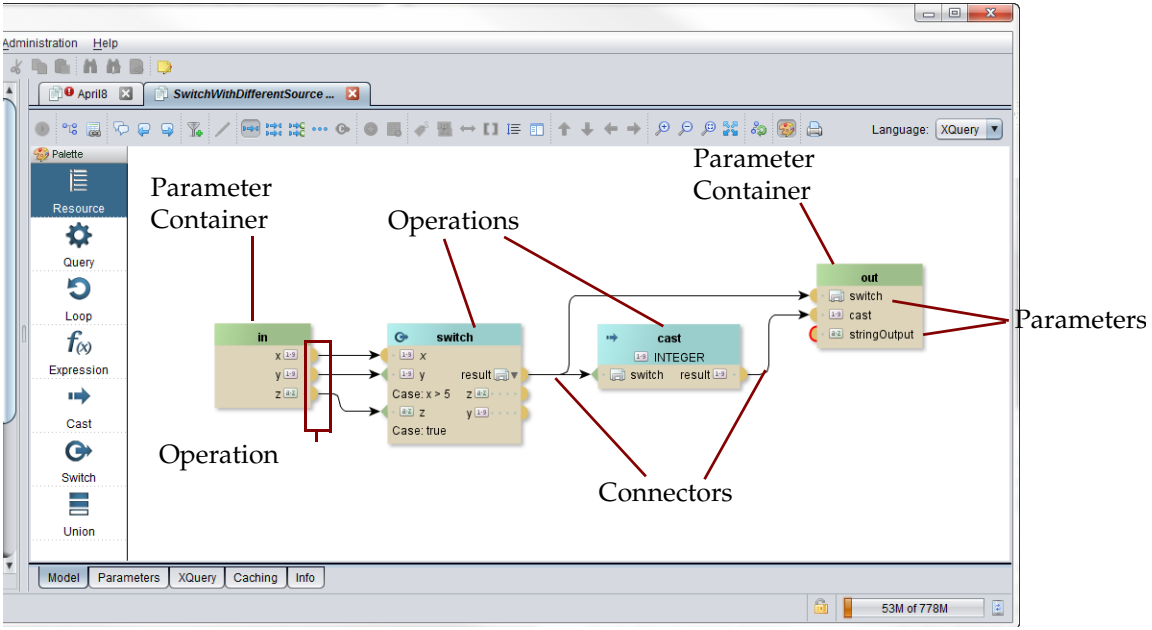
The Transformation Editor Window

The following graphic identifies the main components of the Transformation Editor window.



Transformation Editor Terminology

The following graphic identifies some of the detailed components of the Transformation Editor window.



About Transformation Editor Operations and Parameter Containers

Each of the following Transformation Editor operations can be used to define the logic associated with the transformation:

Name	Description
Cast	Use to convert one data type to another data type.
Expression	Use to define an expression that can be used to compute results.
in	Container for the input parameters for the transform. This container can be empty and not connected to any object on the workspace.
Loop	Use to generate target sequences based on a source sequence.
out	Container for the transformation outputs. This container cannot be empty.

Name	Description
Query	Use to join, filter, and group data.
Resource	Use to add a table, view, XML file, SQL script, or other valid TDV resource to the Transformation editor.
Switch	Use to conditionally map a target from one of a set of sources. You can have any number of inputs, and one output. Each input has an expression associated with it. The first expression that evaluates to true causes its associated input to be passed through as the output.
Union	Use to combine the result sets of two or more items in your transform. All the source inputs must have the same data type. Unions can be used to remove duplicate rows between the items that are input.

Transformation Editor Limitations

There are limitations when using the Transformation Editor:

- Loop chaining is not supported.

  A loop iterator that contains a sequence cannot be used as the source of another loop. To work around this:

  — Instead of multiple loop operations, use a single loop with multiple iterators.

  — Map the child sequence to a child transformation and perform the required secondary looping within that child transform.

- XML Limitations

  — XML substitution groups are not currently supported.

  — XML type restrictions are not supported. The Transformation Editor can use XML types with restrictions, but the restrictions are currently ignored.

  — XML `any` and `all` elements and groups are not fully supported.

  — Some TDV procedures output XML without a schema. The ability to use such procedures might be limited.

  — The XML File, Web Service, and XML/HTTP data sources automatically generate or provide the ability to associate the data source's XML definitions set with the signatures of the resources within those data sources. This allows the Transformation Editor to operate using the XML schemas associated with these resources. If your existing data sources do

not have the associated definition set, you can try re-introspecting or recreating them to get it.

— XML unions (element-based) are not supported. However, XML choices are supported and appear as a canonical Union type within the Transformation Editor.

- Code Generation
  - — Query and Join hints are only applied to queries that will be optimized into SQL.
  - — SQL push optimizations are only applied to query operations and the sources they directly depend upon.
  - — There is no indicator of whether or not a query will be generated as XQuery or SQL. You will need to look at the generated source code to see whether it was optimized or not. In general, if a query's sources are relational, SQL will be generated.
  - — No runtime schema validation is performed.
- Other Limitations
  - — There is no technique to perform element selection. Transformations that make decisions based on element names are not supported.
  - — Cross joins are not supported.
  - — There is no way to handle case-sensitivity mismatches.
  - — There is no literal format for durations.
  - — The cast operation is not supported for casting to a complex structure. If you need to cast the children of a structure, you must assign the structure field by field and insert casts as appropriate for each of the fields. If the structure is a sequence, you can insert a loop operation into your mapping to iterate over all the elements and then provide the individual assignments (with appropriate casting) to the target structure.
  - — There are several SQL functions that do not work well. These include:

    CAST, CORR, COVAR_SAMP, STDDEV, STDDEV_POP, STDDEV_SAM, SUM_FLOAT, VAR_POP, VAR_SAMP, VARIANCE, VARIANCE_POP, VARIANCE_SAMP, XMLATTRIBUTES, XMLFOREST, XMLQUERY

  - — Creating placeholders for intermediate values is not allowed. The workaround is to invoke a child transformation and use the outputs of that transformation as a reference for the intermediate values you want to reuse.

Invocations of child transformations are optimized to avoid the overhead of a full TDV procedure invocation.

— If you rename an element that is directly based on an XML element, the transformation may not work. Avoid renaming element based on XML elements. When you create an element based on an XML element, it normally gets a default name and namespace that matches the XML element. This can be fixed by renaming the element to match the native XML element name.

Using the Transformation Editor

This topic covers basic usage and some common use cases for the Transformation Editor. Transforms can be used to convert data from one type of structure to another, to insert calculations, and to otherwise change the data in ways that are typical for data processing and analysis.

Data can be transformed between relational and hierarchical structures.

This section describes how to perform basic actions within the Transformation Editor:

Use-case tutorial that describes using the Transformation Editor to combine the various elements to create a procedure are included in the *TDV Tutorials Guide*.

Creating a New Transform

Transforms, like procedures, are resources that are available from the Studio resource tree.

To create a new any-any transformation

1. Select a folder in the Studio resource tree.
2. Right-click and select New Transformation, or select File > New > Transformation.
3. Select **Any-Any Transformation**.
4. Click **Next**.
5. Type a name for the transformation.
6. Click **Finish**.

   When you click Finish, the transformation is added to the resource tree, and the Transformation Editor opens in the right pane.
7. To add items, see the instructions in
   - — Editing Operations in the Transformation Editor Model, page 343
   - — Designing a Query Operation, page 355
   - — Adding Expression Operations, page 354
   - — Adding Cast Operations from the Palette, page 357
   - — Adding Union Operations, page 358
   - — Adding Switch Operations, page 359
8. To draw lines between the items, see the instructions in Adding Assign Links, page 346 or Adding Loop Links, page 346.
9. Right-click items on the editor to get a menu with available actions. Most items have:

Copy	Add Parameter	Edit	Change Type
Rename	Change Facets	Delete	Cut

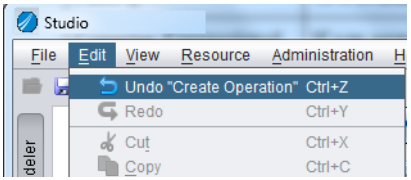
10. Save your transform.

Undoing and Redoing Actions in the Transformation Editor

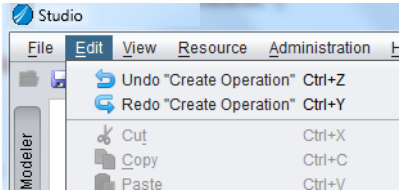
The Transformation Editor provides undo and redo capability.

To undo and redo actions in the Transformation Editor

1. Open your transform.
2. Edit your transform as necessary.
3. To undo a previous action select Edit > Undo <action> from the Studio menu.



4. To redo an action, select Edit > Redo <action>.



5. Save your transform.

Zooming and Arranging the Model

The Transformation Editor view and certain operations on the model can be expanded and collapsed. Searching the model for specific items is also possible. Selections made for zoom and positioning of operations on the Model tab are temporary only. Studio regularly attempts to auto-arrange the view on the Model tab.

To auto-arrange things on the Model

1. Open a transform.
2. Click Refresh Layout.

To zoom the model

1. Open an existing transform in the Transformation Editor.
2. Use the following zooming tools to expand or collapse your transform model:

   — Zoom In
   — Zoom Out
   — Zoom to Fit Contents
   — Full Screen

   Your mouse scroll wheel can also be used to zoom in and out.
3. Continue working with the model.

To grab and pan things in the Transformation Editor Model tab

1. Right-click and drag anywhere in your model.

To move things in the Transformation Editor Model tab

1. Click an operation to select it.
2. Click it again and then drag it to the location where you want it.
3. Arrange your other items.
4. Continue working with the model.

Editing Operations in the Transformation Editor Model

Each of the following Transformation Editor operations has editors that can be used to define the logic associated with the transformation:

Resources	Queries	Loops
Switches	Expressions	

When writing expressions, use the inputs of the operation that contains the expression.

To update operations in the Transformation Editor Model tab

1. Open your transform.
2. Double-click an operation on the Model tab to select it.

   Depending on the type of operation you have selected you should get one of the following editors:

Operation Type	Editor	Description
Expressions	Expression Operation Editor	Provides the name given to the expression and an editable text field that you can use to type your valid expression syntax.
Loops	Loop Iterator Editor	Provides the name given to the loop. You can add levels, define item filters, and use the arrows to arrange the nesting of the levels.
Queries	Query Editor	Provides a complex visual modeling tool with three tabs to help you define the query used within your transform.
Resources	Invoke Resource Operation Editor	Provides the name of the resource, the path to the resource in Studio, and a button that can be used to navigate through a Studio resource tree of valid resources for selection.
Switch	Switch Case Editor	You can use the Switch operation with one or more pairs of expressions. The Switch function evaluates each pair of expressions, and returns the result corresponding to the first boolean expression in the list that evaluates to True.
Cast	Choose Data Type	You can use the CAST function converts a value from one data type to another.
Union		There is no editor for this operation. The operation is created with two inputs and one output that you can use to connect the items that you want to union. Use to combine the result sets of two or more items in your transform. All the source inputs must have the same data type. Union removes duplicate rows between the items that are input.

3. Use Close, OK, or other mechanisms to close the editors when you are finished editing.

4. Save your transform.

Adding Resources

The Transformation Editor can map and transform data to and from relational, XML, or scalar data structures.

You can browse to and drag existing resources defined in TDV into the Transformation Editor.

Note: The XQuery and XSLT procedures cannot be used as resources in the Any-Any Transformation Editor.

To add resources to the Transformation Editor Model tab

1. Open a transform.
2. Locate the resource that you want to add from the Studio resource tree.
3. Click-and-drag that resource onto the Transformation Editor workspace.

Or

1. Open a transform.
2. Click-and-drag the Resource palette icon onto the Transformation Editor workspace.
3. Right-click it and select Edit Resource.
4. Click Choose Resource or type the full Studio resource tree path to the resource you want to use.
5. Optionally, from the Select Resource window navigate to and select your resource, then click **OK**.
6. Click **OK**.

Working with Connections and Loops

Connections and loops help define how data within the transform is processed. This section includes the following topics:

- Adding Cast Operations to Connection Lines, page 351

Adding Assign Links

Links are arrows that you draw between resources to indicate direction of the data transformation. Assignments are used to map data movement between sources and targets.

To add assign links in the Transformation Editor Model tab

1. Select the Assign Link Mode. Alternatively hold SHIFT key while creating a link (Shift+Minus) Transformation Editor toolbar icon.

2. Click an operation handle, while holding your mouse button down, drag the line to the row you want to connect it to.
3. Save your transform.

Adding Loop Links

Loop links add more operations to your Transformation Editor model to make it easier for you to define iterations through sequences. If one of the endpoints of a loop link is a loop, a new loop operation is created on the Model tab.

To add a loop between two operations in the Transformation Editor Model tab

1. Select the Loop Link Mode. Alternatively hold SHIFT key while creating a link (Shift+Minus) Transformation Editor toolbar icon.

2. Click an operation handle, and while holding your mouse button down, drag the line to the row you want to connect it to.
3. Save your transform.

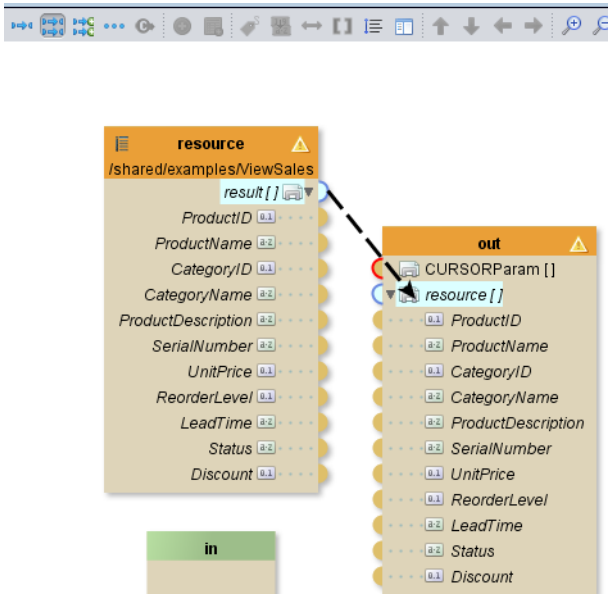
   A new box is drawn on the model to represent the loop. Use this new operation to define the behavior of the loop logic.

Using Auto Map Link Mode

Auto map is a quick way to assign table or view metadata from one operation to another in your transformation. The auto map mode compares the names of the parameters and maps the values that match each other.

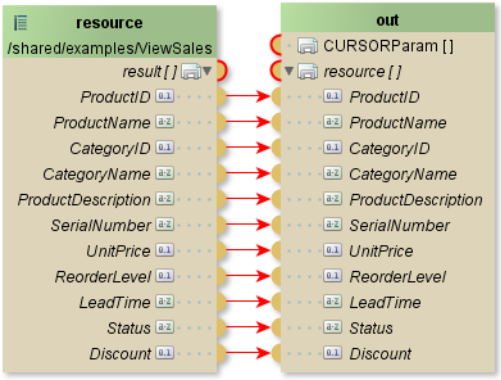
To use the auto map feature

1. Open an existing or create a new transform.
2. Make sure that your transform has at least two operations or resources that contain data structures.
3. Select Auto Map Link Mode.
4. Click and drag the mouse to draw a line between two operations with similar parameters.



5. Release the mouse to connect the two operations.

TDV maps the two structures according to the names that match. For example:



6. Keep or delete the mappings as needed for your transformation.
7. Save your transform.

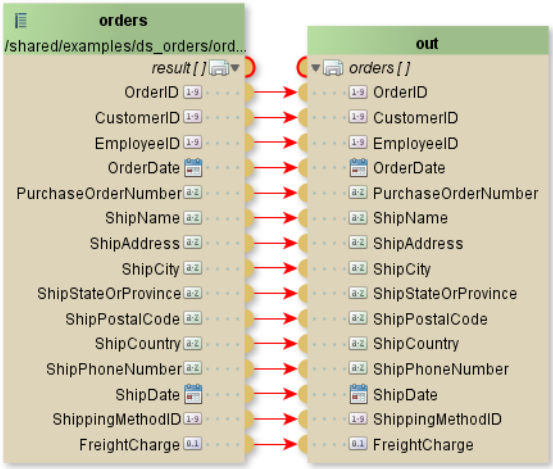
Using Auto Fill Link Mode

Auto fill can be used to assign and replicate a metadata structure from one operation into another in your transform. If the target operation of your auto fill action does not have the parameters that are part of the source operation, they are added to the target operation.

To use the auto fill feature

1. Open an existing or create a new transform.
2. Make sure that your transform has at least two operations or resources that contain data structures as they exist for view, table, or XML schemas.
3. Select Auto Fill Link Mode.
4. Click and drag the mouse to draw a line between two operations with similar parameters.
5. Release the mouse to connect the two operations.

TDV maps the two structures according to the names that match and adds any extra parameters to the target of the auto map. For example:



6. Keep or delete the mappings and parameters as needed for your transformation.
7. Save your transform.

Editing Loop Operations

Loop operations make it easier for you to define looping references and transformations. Edit the loop to define the behavior of the loop logic. When editing, consider the following limits:

- Loop operations are used to generate sequences.
- The size of the target sequence will match the source sequence.
- The content of the target sequence does not necessarily need to match the source.
- Only the size of the target sequence is controlled by the loop.
- Item filters can be used to limit the size of the target sequence.

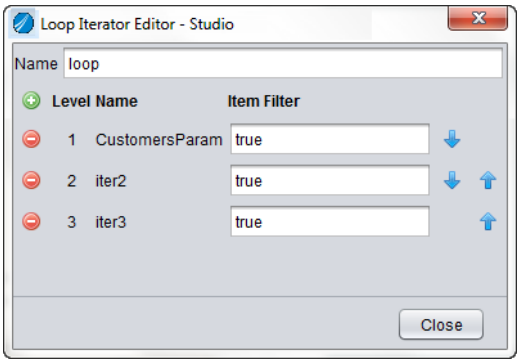
SQL functions can be used in the Query operation if the query gets pushed to a SQL database for execution. Outside of the Query operation you can use canonical, XQuery or custom functions.

If your loop operation contains an expression that contains illegal characters you must enclose them in double quotes. The following characters are considered illegal:

```
'\\', '[', ']', '-', '(', ')', '.','^','$','+', '*','?', '{', '}',
'|', ':', '<', '>', '<', '='
```

To edit a loop between two operations in the Transformation Editor Model tab

1. Double-click to select a loop.



   Depending on what iterator rows you have connected, the loop editor might have one or more iterator levels already defined.

2. To add a new iterator, click the green circle with a plus symbol in it.

   A Name is the name given to the iteration level of the loop which matches the name of the source input of the loop. The Level indicates the nesting level associated with an iterator.

3. To define the filter logic, type text into the Item Filter Field. The item filter is processed every iteration.

Value	Description
True	If the item filter evaluates to true, then the current loop entries used to generate a target sequence item. If the item filter is true, then the target sequence will contain as many items as in the source sequence.
False	If it evaluates to false, it will be skipped.
Complex Expressions	If you provide a more complex expression, output is constrained to data that meets the criteria specified. For example, for totalPrice > 50.0, the loop only outputs entries where the total price is greater than 50. It is only valid for expressions to refer to the input of the loop.

4. Use the arrows to the right of the Item Filter fields to arrange the Filter Name fields in the order that makes sense for the given loop logic that you need to define.
5. Click Close to save your definitions and close the Loop Iterator Editor.
6. Save your transform.

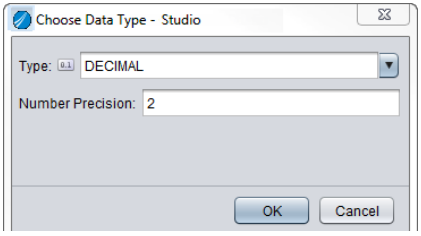
Adding Cast Operations to Connection Lines

While designing your transform, you might notice that some of the link lines are red or orange, which indicates that there is some error or warning message associated with them that needs fixing. Often the warning message can be fixed by adding a CAST function between the two connected points.

To add cast operations to connection lines

1. Open your transform.
2. Select a connection that is displayed in orange or red.
3. Right click and select Insert Cast.
4. From the Transformation Editor palette, select Cast.

   The Cast operation is added to the transform.
5. Double-click the Cast operation.



6. Select the data type that you want your data to become.
7. Specify any additional data type details.
8. Click **OK**.
9. Save your transform.

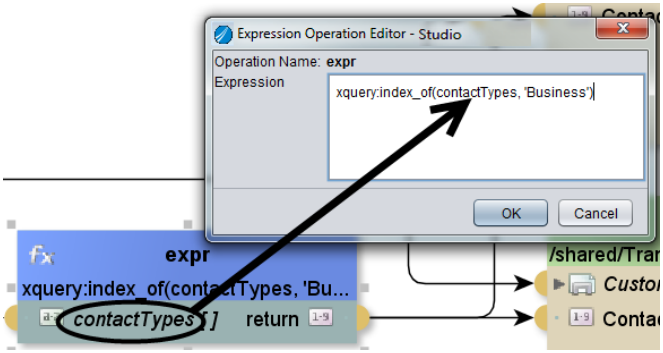
Working with Expressions

Expressions are so commonly used during the definition of an Any-Any Transform, that this topic has been given extra importance so that you will see it.

About Expressions

An expression is a combination of one or more values, operators, and functions that resolve to a value, like a formula. You can use expressions to return a specific set of data. The Transformation Editor can accommodate expressions in queries, loops, switches, and in the stand-alone Expression operation.

When writing expressions, use the inputs of the operation that contains the expression. For example if the operation has an EMP input column named, when you are defining the expression for that operation you can use the EMP input column as a symbol within the expression.



Because of the flexibility required by the Transformation Editor, the expression syntax that is supported is also flexible. Valid expressions can contain operators, names, paths, and literal values. The following table can be used to give you ideas for what you might want to use when defining your Expression.

Expression Components	Description
Operators	Expression operators are used to compute values, for example, with + or -. Operators contain one or two arguments. The supported operators are: NOT, AND, OR, <=, >, <, <>, >=, =, +, *, -, /, and function calls

Names	Valid syntax for names is:
	`[ {namespaceURI} |  prefix: ] name`
	Examples:
	`"customer", "{http://biz.com}customer", "biz:customer"`

Expression Components	Description
Paths	Paths can be used to refer to hierarchical elements of input parameters. Paths are names separated by slashes (/).
Literal Values	Literal values for use in expressions include: • boolean: true | false • integer:[0-9]+ • nil: null • string: 'text' • decimal: [0-9]+.[0-9]* • hexadecimal: 0x[0-9a-f]* • datetime:yyyy-mm-dd [t-hh:mm] | z]
Function Calls	The Transformation Editor makes use of the following categories of functions: • Canonical—A function type that can be used regardless of the target language. • SQL—A function type that can be used only within SQL code. Queries can be generated into SQL. Use the **`sql:`** prefix to specify a SQL function in an expression. • XQuery—A function type that can be used only within XQuery code. Most operators are generated into XQuery. Use the **`xquery:`** prefix to specify a XQuery function in an expression. • Custom—A function type that is defined as custom within TDV. Use the **`custom:`** prefix to specify a custom function in an expression. The following canonical functions are available for use within expressions: • Aggregate: AVG, MIN, MAX, SUM, COUNT • Character: CONCAT, SUBSTRING, UPPER, LOWER, LENGTH, TRANSLATE, REPLACE, MATCHES, CHARACTER_LENGTH • Numeric: ABS, CEIL, FLOOR, ROUND • Date: CURRENT_DATE, CURRENT_TIME

Canonical functions are built in and cannot be created.

When creating a custom function, the Transformation Editor follows the same rules as for TDV SQL. You create a procedure with one output and promote it as a custom function within the TDV. Using it within a transform, requires the use of the "custom" prefix. For example, if you made a custom function called "amazing", then you'd invoke it within a transform expression using "custom:amazing(x,y,z)".

XQuery functions must be invoked with the "xquery" prefix. For example, to invoke the XQuery concat function, you would use "xquery:concat()". XPath expressions are not supported.

Transform expressions don't allow dashes within symbol names. Replace dashes (-) with underscores (_).

Adding Expression Operations

Most operations have an editor that allows you to define expressions.

To transform your data using Expression

1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Transformation Editor palette, select Expression.
3. Click and drag the mouse anywhere in the Model tab.
4. Double-click the Expression operation that you just added.
5. Use the editor to specify your expression syntax. For example, type syntax similar to any of the following:
   - salary < 50000
   - EMP > 34
   - {http://biz.com}customer/biz:orderID < 900 AND NOT {http://biz.com}customer/biz:customerID = 20
   - Sales/finalsale > 1000

   If your expression contains illegal characters you must enclose them in double quotes. The following characters are considered illegal:

```
'\\', '[', ']', '-', '(', ')', '.','^','$','+', '*','?', '{', '}',
'|', ':', '<', '>', '<', '='
```

6. Save your transform.

Working with Operations

Operations provide the main processing functionality to transform your data. This section includes the following topics:

Designing a Query Operation

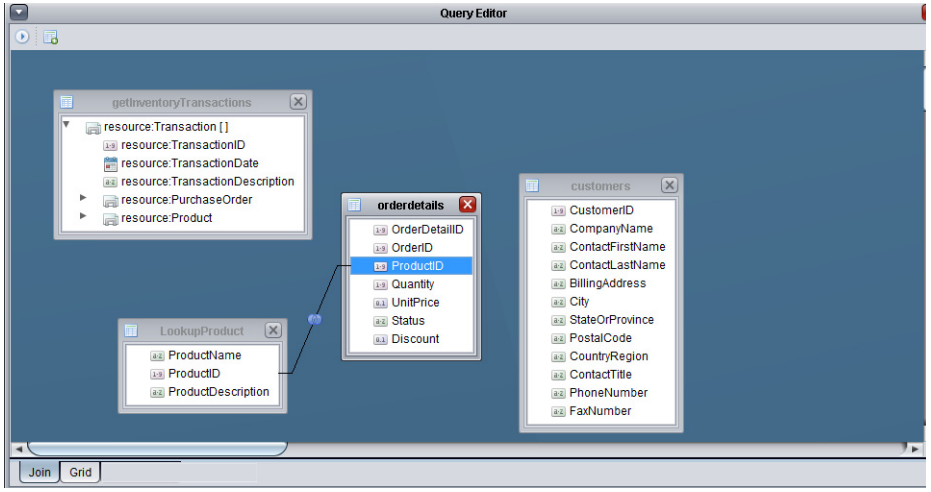
The Query editor that can be accessed through the Transformation Editor is similar in behavior to the Studio View editor. Queries allow data to be joined, sorted, grouped, filtered, and projected.

SQL functions can be used in the Query operation if the query gets pushed to a SQL database for execution. Outside of the Query operation you can use canonical, XQuery or custom functions.

To transform your data using query

1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Transformation Editor palette, select Query.
3. Click and drag Query anywhere in the Model tab.
4. Double-click the Query operation to obtain a Query editor.
   - On the Join tab of the editor, you can click and drag resources from the Studio resource tree. You can also draw relationship lines between the rows

of the resources that have been added. For more information on similar functionality, see the View editor documentation in the *TDV User's Guide*.



— On the Grid tab you can:

— Specify columns for projection.

— Move the columns up or down to determine their output order.

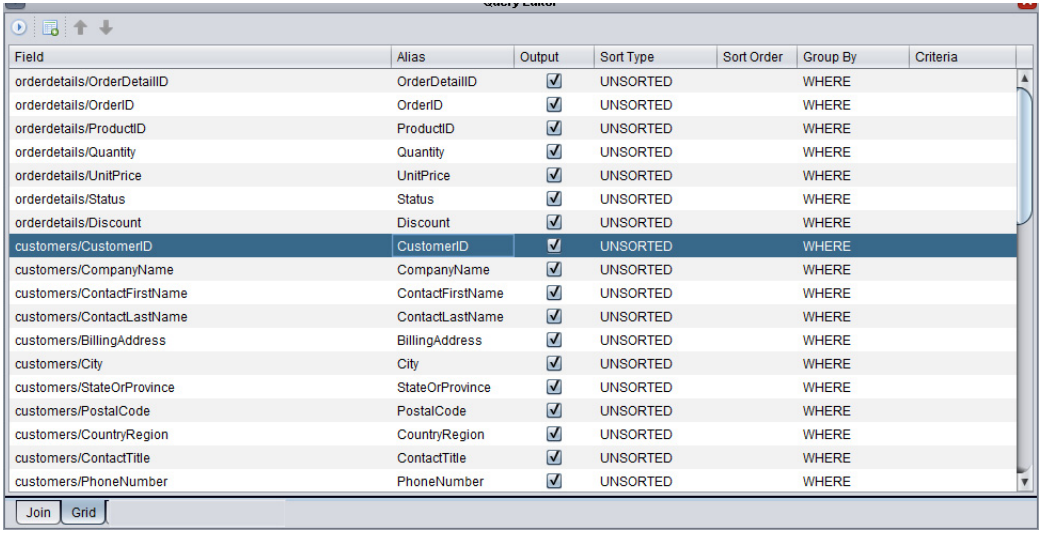
— Supply aliases for column names.

— Sort columns in ascending or descending order.

— Specify the **GROUP BY** option.

— Specify query constraints in the **WHERE** or **HAVING** clause of the query.

— Add functions and declare variables for the query.



5. Save your transform.

Adding Cast Operations from the Palette

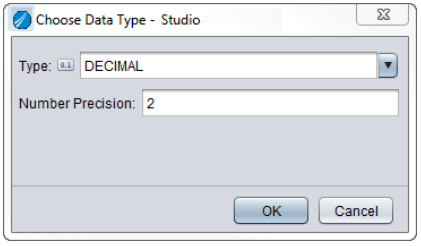
You can use the CAST function converts a value from one data type to another.

A cast operation is always required when performing a narrowing conversion, such as converting from an xs:string to a VARCHAR.

To transform your data using cast

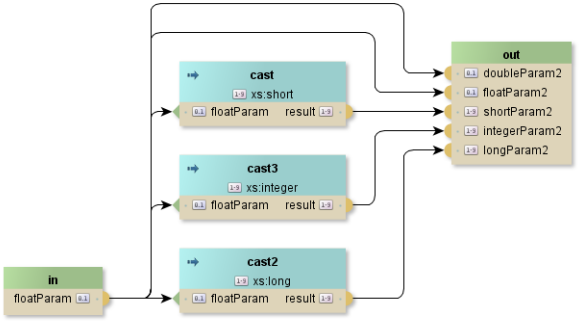
1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Transformation Editor palette, select Cast.
3. Click and drag the mouse anywhere in the Model tab.
4. Click to select the Cast operation that you just added.
5. Draw a link from the column that you want to transform with the cast function, to the CAST operation.
6. Save your transform.

7. Double-click the Cast operation.



8. Select the data type that you want your data to become.
9. Specify any additional data type details.
10. Click **OK**.
11. Connect the result side of the Cast operation with the column where the data will be consumed next.
12. Save your transform.

    The following model diagram shows a transform using Cast operations:



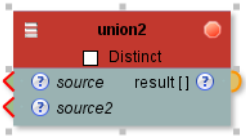
Adding Union Operations

Union functionality provides the ability to combine data into one unified data set. You can specify whether or not to filter out duplicates.

The sources for a union must have exactly the same structure (element names and types) in order for them to be combined using a union operation. The two different spellings of "flag" and "falg" will produce an error. You can create a child transformation to act as an intermediate value or as a structure cast.

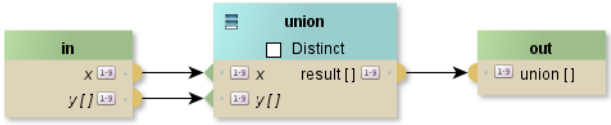
To transform your data using union

1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Transformation Editor palette, select Union.
3. Click and drag the mouse anywhere in the Model tab.



4. Connect at least two parameters from somewhere else in your transform to the Union operation. The two parameters must be of the same data type.
5. Connect the Union operation's result to another operation within your transformation.
6. (Optional) Add more source parameters by:
   a. Click the Union operation to select it.
   b. Right-click and select **Add Source** from the menu.
7. (Optional) Select **Distinct**, to indicate that you only want unique values in the outcome of the Union operation.
8. Save your transform.

The following model diagram shows a simple transform using the Union operation:

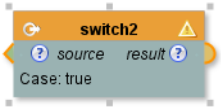


Adding Switch Operations

You can use the Switch operation to conditionally control the flow of data from one or more sources. Each source has a corresponding case expression. At runtime, the case expressions are evaluated from top to bottom. The first case expression to evaluate to true causes the corresponding source to be output from the switch.

To transform your data using switch

1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Transformation Editor palette, select Switch.
3. Click and drag the mouse anywhere in the Model tab.

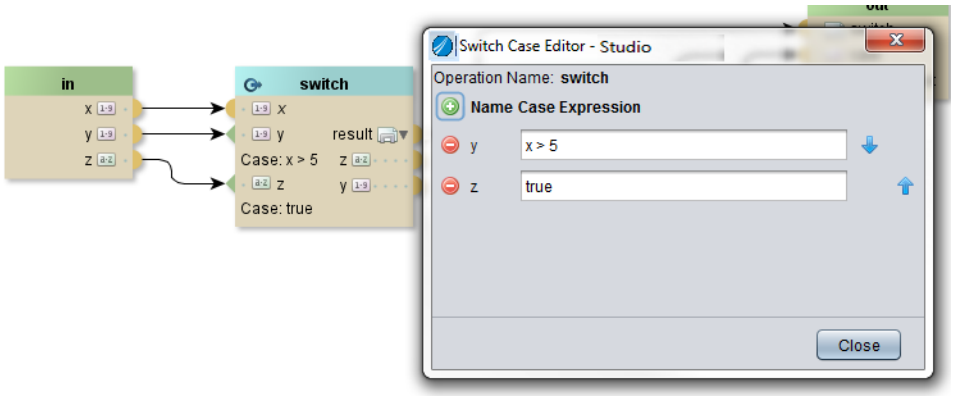


4. Connect at least one parameter from somewhere else in your transform to the Switch operation.
5. Connect the Switch operation's results to other operations within your transformation.
6. (Optional) Add more source parameters by:
   a. Click the Switch operation to select it.
   b. Right-click and select **Add Source** or **Add Parameter Insert** from the menu.
7. (Optional) Rename source or result parameters by:
   a. Select the operational handle of the parameter you want to rename.
   b. Right-click and select **Rename "<parm_name>"**.

   Or, click Rename from the Transformation Editor Model tab toolbar.

   c. Type the new name.

8. Define the conditions on the switch. Each condition is called a **Case**. Each Case is evaluated in the order specified and the first case to evaluate to true is returned in the result.



   — To add new Case expressions, click the green plus button.

   — To edit the expression defined for each case, click in the text edit fields and edit the text as necessary to define your expression.

   — To adjust the order that the conditions are evaluated, use the arrow buttons to move the row up or down.

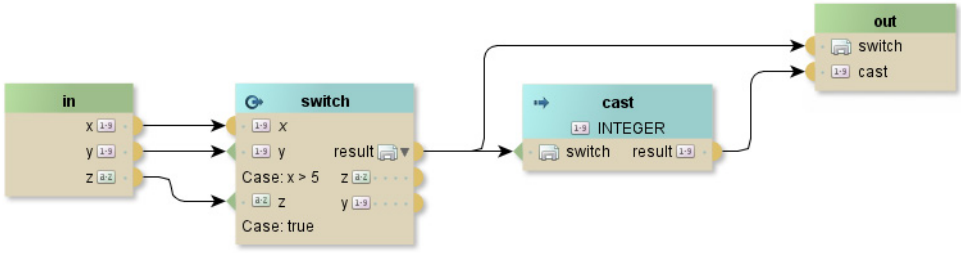
   — To close and save your definitions, click **Close**.

   If your expression contains illegal characters you must enclose them in double quotes. The following characters are considered illegal:

```
'\\', '[', ']', '-', '(', ')', '.','^','$','+', '*','?', '{', '}',
'|', ':', '<', '>', '<', '='
```

9. Save your transform.

   The following model diagram shows a transform using the Switch operation:



Using the Create Operation Button

To use the Create Operation button

1. Open a transform.
2. Click Create Operation.

   The last operation that was selected or is currently selected on the palette is added to the transform workspace.
3. Save your transform.

Working with Operation and Parameter Container Details

Most operations contained detailed elements that can be edited or configured to customize the behavior of your transform. This section includes the following topics:

- Defining Element Facets, page 362
- Defining Constant Values, page 363
- Editing Import Paths, page 364
- Edit Namespace Prefix Maps, page 366
- Using Cycle I/O Direction to Add or Delete Parameters, page 368

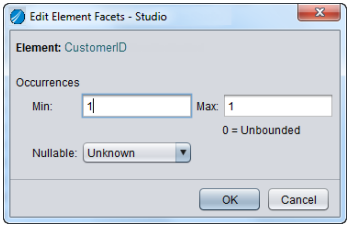
Defining Element Facets

Element facets are the details defined for each of the elements being defined for the XQuery transformation code. They are primarily responsible for setting minimum and maximum occurrences, and indicating if the value can be NULL or not.

Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.

To define element facets

1. Access the Element Facets Editor in several ways including:
   — From the right-mouse menu of an operation handle, by selecting Change Facets.



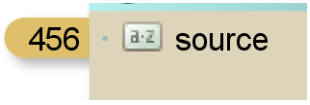
2. Review and modify the values for Min and Max occurrence.
3. Determine the setting you want specified for Nullable:
   — Unknown
   — True
   — False
4. Click **OK**.

If the min and max values were changed, they are displayed next to the parameter name.

Defining Constant Values

Depending on the structure and content of your transform, you might want certain parameters to always evaluate to the same value. This constant value must be a valid value for the data type of the parameter for which you are defining it.

After the value is defined, it will display in the operation handle similar to the following:



Which would evaluate similar to the following in the generated XQuery:

```
element source
{
456
}
,
```

To define constant assignments

1. Access the Constant Assignment edit field in several ways including:
   - — From the right-mouse menu of an operation handle, by selecting Create Constant Assignment.
   - — From the Transformation Editor Model tab toolbar.

   The operation handle populates with a value of zero.
2. From the right-mouse menu, select one of the following:
   - — Edit
3. Type the value that you want used as the constant. For example:
   - — 50000
   - — 34
4. Save your transform.

Editing Import Paths

Import paths allow you to point the Transformation Editor at a set of one or more definition sets that are defined within TDV. These definition sets are then used by the Transformation Editor to provide you with a set of types that you use for parameters. By default, the Transformation Editor points to the following paths:

- /lib/types/sql
- /lib/types/extendedSql
- /lib/types/xmlSchema2001

The paths control what types are available when adding parameters or changing their type.

If your transform uses type definitions from several definition sets and two or more of the definition sets contain an element with the same name, the first occurrence of it is the one that controls the element definition. The order that the definition sets is read can be edited.

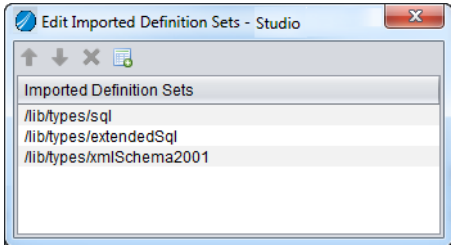
To add a definition set to the list of import paths

1. Open your transform.
2. From the Studio resource tree, click and drag a definition set into the Transformation Editor. Or, add a new parameter and browse to the definition set.
3. Save your transform.

To edit the list of import paths

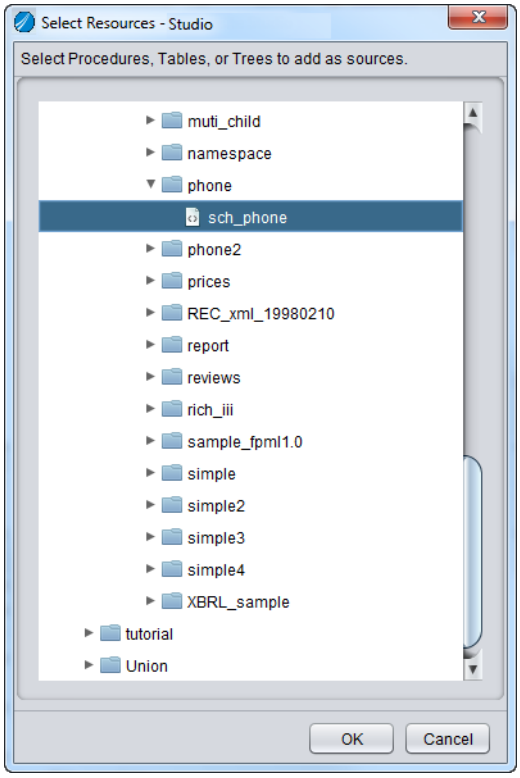
1. Open your transform.
2. Select Edit Import Paths from the toolbar.

   The Edit Imported Definition Sets editor opens and displays the list of paths currently available to the Transformation Editor.



3. Click Import Paths to obtain a selection window that you can use to navigate through a list of valid Studio resources available to add as an imported definition set.

4. Use the Select Resources dialog to select the additional definition set that you want added to the Imported Paths list.



5. Click **OK**.
6. Determine if the order of the list should be changed and if any included paths need deletion.
   - To reorder paths, select a path and use the up and down arrows.
   - To delete a path from the list, select the path and click the red X.
7. Close the Edit Imported Definition Sets dialog.
8. Save your transform.

Edit Namespace Prefix Maps

Namespaces are used to avoid element name conflicts. Because element names are defined by the developer, there can be conflicts when code from more than one developer is combined. The name conflicts in XML can be avoided using a name prefix.

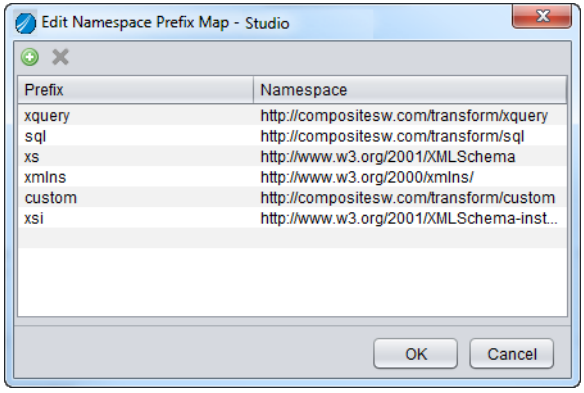
The XML Namespace represents the root URL path that is used for XML-based names. The Namespace is augmented with the specific names of things that are then created.

For example, http://masterschool.com/transform could be the root path assigned to your specific business_school_transform, which would result in the following:
`http://masterschool.com/transform/business_school_transform`

To edit the namespace prefix maps

1. Open your transform.
2. Select Edit Namespace Prefix Maps from the toolbar.

   The editor opens and displays the list of prefixes currently used by the Transformation Editor.



3. To add a new prefix, use the green arrow button to create a new Namespace prefix.
4. To edit an existing, double-click in the cell that you want to edits and begin typing to edit the namespace text.

   If you changed the text in the Prefix column and that name had been shown as part of your transform, the new name should now appear in your transform on the Model tab.

   If you changed the text in the Namespace column, the actual namespace that is used when the transform is executed is changed.
5. Click **OK** to save your changes.

To view the namespace prefix maps declared for your transform

1. Open your transform.
2. Click the XQuery tab.
3. In the XQuery declarations section, look for text similar to the following:

```
declare namespace xquery =
"http://compositesw.com/transform/xquery";
declare namespace sql =  "http://compositesw.com/transform/sql";
declare namespace ext2 =
"http://www.compositesw.com/extensions2";
declare namespace xs =  "http://www.w3.org/2001/XMLSchema";
```

Using Cycle I/O Direction to Add or Delete Parameters

The Cycle I/O Direction button can be used to add parameters to the in and out containers.

To use the Cycle I/O Direction button

1. Open a transform.
2. Select the in or out container.
3. Right-click and select Add Parameter.
4. Specify a data type for the parameter and name the parameter.
5. Click the Cycle I/O Direction button.

   The parameter is moved to the operation that it wasn't in.
6. Optionally, click the button again to add the parameter in both operations.
7. To delete the parameter, select the parameter and then delete it.

   If a parameter was added to the in and out containers using the Cycle I/O Direction button, deleting the parameter from one of the operations deletes it from both operations.
8. Save the transform.

Deleting Operations in the Transformation Editor

In and out are not considered operations and cannot be deleted. Even if there are no parameters in the in container, you cannot delete it from the Transformation Editor Model tab.

To delete operations from the model tab of the Transformation Editor

1. Click the operation to select it.
2. Click the delete button on your keyboard or select Delete from the right-mouse menu.
3. Save your changes.

Working with Messages

The Transformation Editor has a lot of built-in validation of the design phase of your transform. Viewing the messages can help you resolve design problems that prevent your transform from executing. By reviewing and attempting to resolve the various messages can be used to improve the quality of your resulting transform.

- Viewing Messages, page 369
- Using Messages to Refine Your Transform, page 370

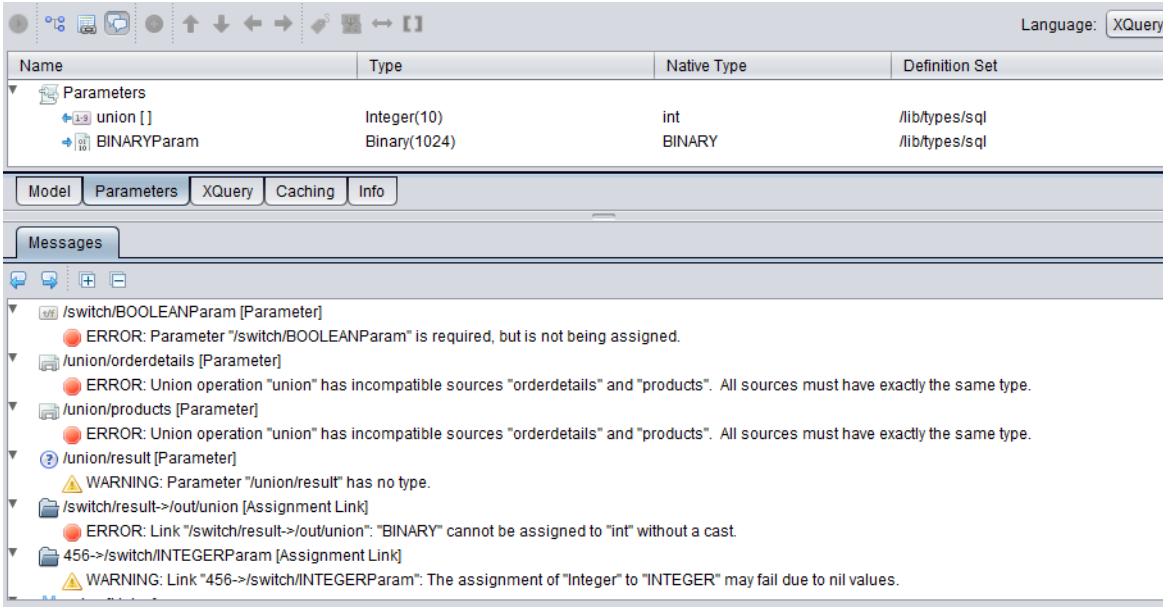
Viewing Messages

To view messages

1. Open your transform.
2. Click Show Messages on the toolbar.

   The messages pane opens in the split panel and displays messages for the transform you are editing. In the following example, the messages button was

selected while on the Parameters tab, but messages can be viewed from other portions of the Transformation Editor.



3. Review the messages.
4. Determine a best course of action to correct your transform.
5. Use the Model and Parameters Tab to implement the necessary changes.
6. Save your transform.

Using Messages to Refine Your Transform

When you select a message in the messenger tab, the affected item is selected on the Model tab. You can use these features of the Transformation Editor to speed your design and deployment of your transforms.

To view messages

1. Open your transform.
2. Click Show Messages on the toolbar.
3. Select a message in the message tab.

   Use the up and down arrow buttons to move up or down the error message list and change the focus in the Model tab.

4. Edit the selected item in the Model tab.
5. Save your transform.

Editing Parameters on the Parameters Tab

The Transformation Editor's Parameters tab displays all of the parameters listed in the in and out containers from the Model tab.

The parameters displayed in the Transformation Editor conveys the cardinality of the parameters in the following ways:

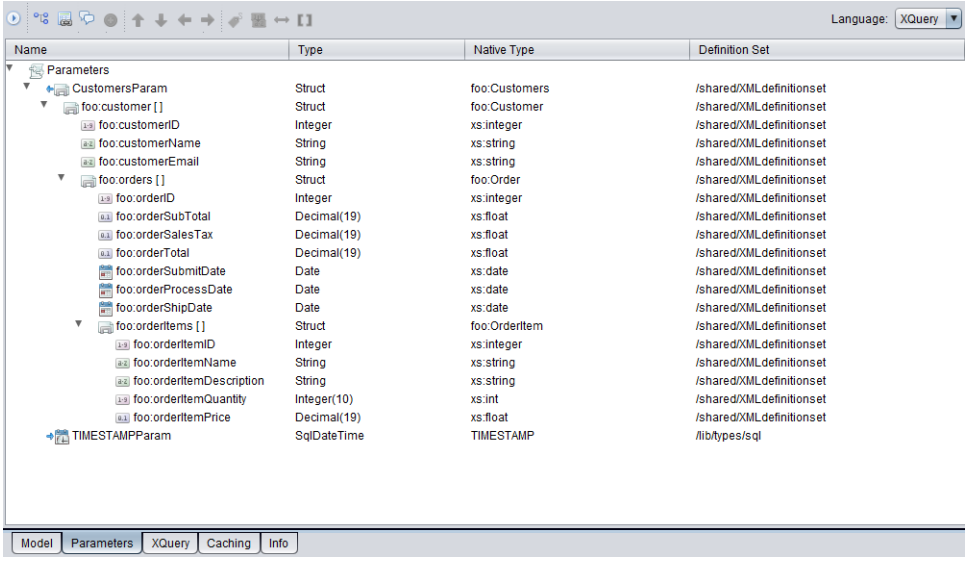
- If the parameter is a sequence, square brackets are shown along with its name. For example, "a[ ]".
- If the max occurs of a sequence is non-zero, that number is displayed within the square brackets. For example, "a[45]".
- If the min occurs of a sequence is non-zero, that number is displayed as a the first part of a range. If max occurs is 0, the range has no upper bound and is displayed similar to "a[4-*]". If max occurs is specified, it is displayed similar to "a[4-45]".

The Parameters tab of the Transformation Editor can be useful as a single parameter editing tool across all of the operations in your transform. It can be more efficient, for example, to use the Parameters tab to change the data type of all TINYINT parameters to INT.

To edit parameters using the Parameters tab

1. Open your transform.
2. Select the Parameters tab.

The following graphic shows several parameters that exist for the in and out containers.



3. Select any row.
4. To add a new parameter, use the green plus button.
5. To adjust the order or position of the parameters use the arrow buttons.
6. Optionally, you can:
   - Rename
   - Change the data type
   - Change the I/O direction
   - Change Facets
7. Save your transform.

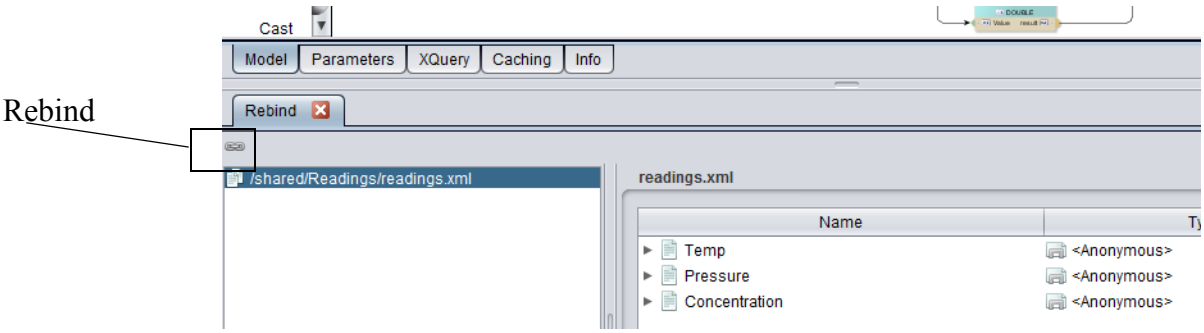
Rebinding Transformation Operations

When importing or exporting information to and from Studio, sometimes the relative location of source data or objects change. For example, when exporting a transform to a CAR file so that it can be used by a colleague, the XML data source might depend on an XML file with data that is stored in C:/temp but is in the D:/users/temp directory on the colleague's computer. For the transform to work on the colleague's computer they must point the transform at the new location for the data by rebinding the transform.

To rebind a transform

1. Open the transform that appears to need rebinding.

Typically, Studio knows that a rebind issue has occurred and displays the Rebind tab for the transform.



2. If you need to modify a data source or other resource to point to the location of the data on your machine, do so.
3. Click the Rebind button.
4. Navigate to the valid location for the resource in your Studio tree.
5. Click **OK**.

Working with the Transformation Editor XQuery Tab

The XQuery tab that is included as part of the Transformation Editor is an excellent tool to use for reviewing the XQuery code that will be generated by the transform that you are defining. After adding a few operations and connections to your transform, you can preview the code to get a feel for how the transform will act when it is executed. As you get more comfortable with the code that TDV will generate for your transform, you can design models that produce the results that you need.

Queries that are part of your transform have generated SQL that you can review. TDV optimizes the SQL so that it can be pushed for execution at the database level.

Topics included in this section are:

- Viewing and Copying the XQuery Code From Your Transform, page 374
- Validating the Transform XQuery Code Using Studio, page 374

Viewing and Copying the XQuery Code From Your Transform

Looking at the code on the XQuery tab of the Transformation Editor can help you understand the code that TDV generates for the transform that you are defining. If the source code generated by the Transformation Editor is inadequate for your unique situation, you can copy the XQuery source of the transform into a newly created XQuery Procedure.

To view and copy the XQuery code

1. Open the Transformation Editor.
2. Save your transform.
3. Select the XQuery tab.
4. Scroll up and down to review the code.
5. Select a portion or all of the code and copy it (using ctrl+c).

Validating the Transform XQuery Code Using Studio

Copy and paste the XQuery code into a different Studio procedure editor to validate that what the Transformation Editor is generating is valid.

This validation method can be used to move your transform code out of the confines of the Transformation Editor code generator so that you can define your transform using XQuery code directly.

To validate the XQuery code that is created by the Transformation Editor

1. Open your transform.
2. Select the XQuery tab.
3. Use your mouse to highlight and select all of the code text.
4. Copy the selected text.
5. From the Studio resource tree, right-click Shared.
6. Select New XQuery Procedure.
7. Name your new procedure.
8. Select all of the text in the XQuery tab and paste your transform XQuery code.
9. Save the procedure.
10. Execute the procedure.
11. If errors are generated, return to your transform and attempt to resolve them using the Transformation Editor.

Using a Transform as a Resource in Another Transform

Reusing transform definitions can speed development of larger more complex data transformation tasks.

To transform your data using nested transforms

1. Create your transform, for instructions, see Creating a New Transform, page 341.
2. From the Studio tree, select an existing transform and click and drag it onto the Model tab.
3. Save your transform.

Caching Transform Data

The Transformation Editor's Caching tab gives you access to set up several of the standard TDV caching features. Because TDV treats transforms as if they were procedures, the caching features that you can enable are the features that are generally supported for other TDV procedure resources. For information on caching with TDV, see TDV Caching, page 465.

The procedure caching process uses one storage table for each output cursor and an additional storage table for any scalar outputs. For example, a procedure with two INTEGER outputs and two CURSOR outputs would use three tables:

- One for the pair of scalars
- One for the first cursor
- One for the second cursor

If a procedure has input parameters, the cached results are tracked separately for each unique set of input values. Each unique set of input parameter values is called a variant.

A procedure cache that uses non-null input parameters must be seeded with at least one variant from a client application other than Studio, for the **Cache Status** to change from **NOT LOADED** to **UP**. Using the **Refresh Now** button does not change the status of a cache that is not loaded. Even if procedure caching configuration is correct, the status does not show that it is loaded until a client seeds the cache.

When a procedure cache is refreshed, all the cached variants already in the table are refreshed. If no variants have yet to be cached, then nothing is refreshed or only the null input variant is refreshed. You can refresh a procedure cache from Studio or using the **RefreshResourceCache** procedure. (See the *TDV Application Programming Interfaces Guide*.)

To define a cache for your transform

1. Open your transform.
2. Validate that the input parameter values for your transformation do not exceeds 255 characters.

   If the parameter exceeds 255 characters, the procedure call is not cached. Instead, it is executed as if caching was disabled. TDV tracks procedure variants by converting the input parameter values are to a string to compare for uniqueness.
3. Select the Caching tab.
4. Click Create Cache.
5. Define the cache.

   See the *TDV User's Guide* for more information on caching concepts and how to define caching on a procedure.
6. Save your transform.
7. Optionally, configure the **Maximum number of procedure variants** field in the **Advanced** section of the **Caching** panel to store the variant result sets produced from your transforms.

   By default, the maximum number of stored variants is 32. If the cache is full, the least recently used variant is swapped out for the latest variant.

Definition Sets

This topic describes definition sets including what they are, how to define them, and how to use them in TDV.

These topics are covered:

About Definition Sets

A *definition set* is a set of definitions you create that can be shared by other TDV resources. A definition set includes SQL data types, XML types, XML elements, exceptions, and constants. You can refer to the resources in a definition set from any number of procedures or views. The changes you make in a definition set are propagated to all the procedures or views using the definition set. You cannot publish a definition set. The definition sets you create are reusable and can be shared with any user with appropriate access privileges in the system.

TDV supports three types of definition sets:

- XML-based
- SQL-based
- WSDL-based

TDV supports these definition sets with the following data types and arrays of those types:
`BINARY, BLOB, VARBINARY, DECIMAL, DOUBLE, FLOAT, NUMERIC, BIGINT, BIT, INTEGER, SMALLINT, TINYINT, CHAR, CLOB, LONGVARCHAR, VARCHAR, DATE, TIME, TIMESTAMP, CURSOR, ROW, XML`

Studio has an editor for building and editing definition sets. The definition set editor opens automatically when you create a definition set. For details, see Creating a Definition Set, page 378. At any time, you can open the definition set in the editor by double-clicking the definition set in the resource tree.

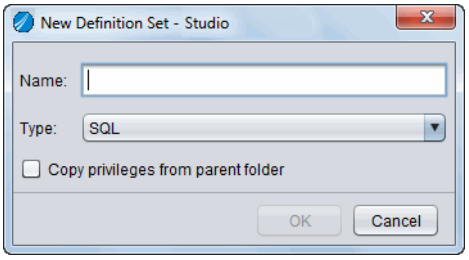
Creating a Definition Set

You can create a definition set anywhere in the Studio resource tree except within Data Services. For other details, see Locating a Container for a TDV Resource, page 40.

This section describes how to create SQL, XML, or WSDL definition sets. An XML definition set can be created from either a schema or a raw XML file instance.

To add a new SQL, XML, or WSDL definition set

1. Right-click at an appropriate location in the resource tree, and select New Definition Set. Or, at the chosen location you can select File > New > Definition Set.



2. In the New Definition Set dialog:
   a. Type a name for the definition set.
   b. Choose a type for the definition set—SQL, XML, or WSDL—in the Type drop-down list.
   c. Click OK.

   The definition set is added to the specified location in the resource tree, and the definition set editor opens in the right pane.

3. Add definitions to the definition set using the instructions for the type of definition set:
   — SQL—See Creating a SQL Definition Set, page 379.
   — XML—Creating an XML Definition Set, page 382.
   — WSDL—Creating a WSDL Definition Set, page 384.

Creating a SQL Definition Set

You can create a SQL definition set that defines parameters, exceptions, and constants. You can then refer to the parameters, exceptions, and constants in a definition set from any procedure or view in TDV, and also share the definition set with any user who has appropriate privileges in the system. You can type the SQL definitions directly, or import existing definitions.

To create a SQL definition set

1. Follow the steps in Creating a Definition Set, page 378, choosing SQL as the type of definition set.

   See SQL Definition Set Editor, page 675 for information about the SQL definition set editor toolbar buttons and panel features.

2. Select the Types tab in the editor.

   You can create type definitions for a SQL definition set in one of two ways:

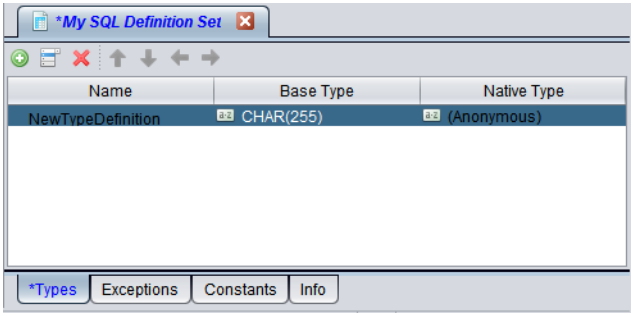
   — You can add new type definitions one at a time.

   — You can upload an existing definition set. You can include both SQL-type definitions and XML-type definitions in a SQL-type definition set.

3. Add one or more new definitions by following these steps:

   a. Click the Add button and select one of the data types from the menu.

Studio adds a definition of the specified type, giving it the default name NewTypeDefinition. For example, if you select String > CHAR, the new definition would look like this:



b. Rename the newly added type definition by clicking in the Name field and typing the new name. Or, you can right-click and select Rename from the menu.

c. If the type is Complex and you have created multiple xmlElements, you can move or indent a type definition by using the navigation arrows in the editor's toolbar.

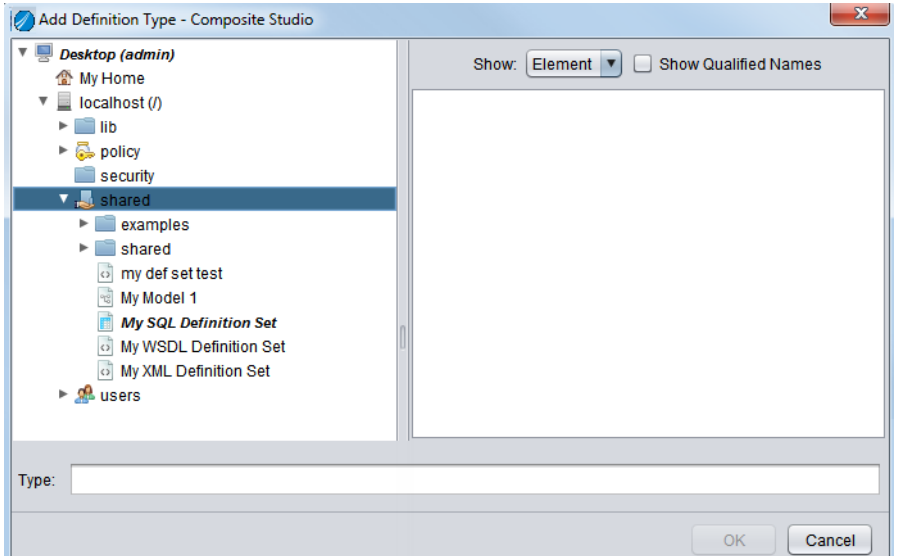
d. Repeat to create as many type definitions as you need.

e. Save the definition set.

4. Optionally, add an existing definition set from the Studio resource tree by following these steps:

   a. Click the Add button and select Browse from the menu.

The Add Definition Type window opens.



b. In the resource tree in the left pane, select a definition set.

c. In the right pane, select the definition set.

The qualified name for the selected definition set appears in the Type field at the bottom if the window.

d. Click OK.

e. Back in the new definition set panel, rename the newly added definition set by clicking in the Name field and typing the new name. Or, you can right-click and select Rename from the menu.

f. Expand the definition set and edit it as necessary. You can:

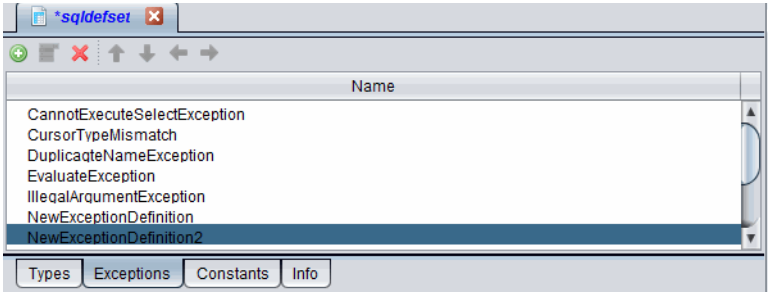
— Use the right-click menu to rename, change the data type or value, or delete a type definition.

— Move or indent a type definition using the navigation arrows in the editor's toolbar.

g. Save the definition set.

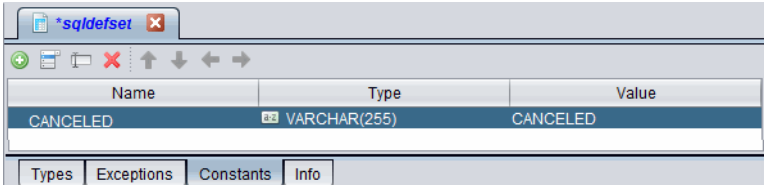
5. To add exceptions, select the Exceptions tab, click the Add button, and add exceptions the same way you added types.

The **Exceptions** example below displays some exceptions available in the system.



6. To add constants, select the Constants tab, click the Add button, and add constants the same way you added types and exceptions.

   The Constants panel with an example constant is shown below.



7. To supply a value for a constant:
   a. Select the constant.
   b. Right-click the Value field and select Change Value.
   c. In the input dialog, supply a value for the constant, and click OK.
8. Optionally, click the Info tab to review information about this definition set or supply an annotation for the definition set.
9. Save the definition set.

You can now reference the SQL definition set and all of the data types contained in the definition set from stored procedures, XQuery procedures, and XSLT procedures. See Using a SQL Definition Set, page 385.

Creating an XML Definition Set

You can create an XML definition set by typing it in or by importing existing XML definitions.

To create an XML definition set from an XML file instance

1. Follow the steps in To add a new SQL, XML, or WSDL definition set, page 378, choosing XML as the type of definition set.

   The XML Schema tab is displayed in the editor that opens on the right. See XML Definition Set Editor, page 675 for information about the XML definition set editor toolbar buttons and panel features.

2. Enter XML schema definitions in one of three ways:

   — Type the XML schema definitions directly into the panel.

   — Import the XML schema definitions from an existing file in your file system as follows:

   a. Click the Import XML Schema Definitions From File(s) button.

   b. In the File Location dialog, type the full pathname to the XML file in the File field, or browse to the XML file that you want to import.

   c. Optionally (in the File Location dialog), check the Show Advanced Options check box to view the connection properties and alternate location mappings for this file.

   d. Click OK.

   — Import the XML schema definitions from an XML Instance as follows:

   e. Click the Create XML Schema Definitions From XML Instance button.

   Studio opens an Open dialog in which you can choose the XML file. This needs to be an XML file, not a schema.

   f. Click Open.

3. In the editor, view and optionally edit the schema.

4. Use the Validate XML Schema Definitions button in the toolbar to validate the schema.

   The validation process parses the XML file into a schema of element declarations and type definitions, which now appear in the XML Schema panel.

   A field in the lower right corner of the panel shows the current line and column position of the cursor (separated by a colon) to help you find errors in the schema if it is found to be invalid.

5. Use the Format XML Schema Definitions button in the toolbar to format the definitions with indentation and font coloring (keywords in orange type and literal definitions in blue type).

6. Save the definition set.

You can now reference the XML definition set and all of the XML types defined in the definition set from stored procedures, XQuery procedures, and XSLT procedures. See Using an XML Definition Set, page 386.

Creating a WSDL Definition Set

You can create a WSDL definition set by typing it in or by importing existing WSDL or XML schema definitions.

To create a WSDL definition set

1. Follow the steps in To add a new SQL, XML, or WSDL definition set, page 378, choosing WSDL as the type of definition set.

   The WSDL tab is displayed in the editor that opens on the right. See WSDL Definition Set Editor, page 677 for information about the WSDL definition set editor toolbar buttons and panel features.

2. Enter WSDL schema definitions in one of these ways:

   — Type the schema definitions directly into the panel.

   — Import the WSDL or XML schema definitions from an existing file in your file system as follows:

   a. Click the Import WSDL and XML Schema Definitions From File(s) button.

   b. In the File Location dialog, type the full pathname to the WSDL or XML file in the File field, or browse to the WSDL or XML file that you want to import.

   c. Optionally, check the Show Advanced Options check box to view the connection properties and alternate location mappings for this file.

   d. Click **OK**.

3. Use the Validate WSDL and XML Schema Definitions button in the toolbar to validate the schema.

   Validating the schema parses the schema into Element Declarations and Type Definitions, which you can find in the XML Schema panel.

   A field in the lower right corner of the panel shows the current line and column position of the cursor (separated by a colon) to help you find errors in the schema if it is found to be invalid.

4. Use the Format WSDL and XML Schema Definitions button in the toolbar to format the definitions with indentation and font coloring (keywords in orange type and literal definitions in blue type).

You can now reference the WSDL definition set and all of the XML types defined in the definition set from stored procedures, XQuery procedures, and XSLT procedures. See Using a WSDL Definition Set, page 387.

Using a SQL Definition Set

You can reference a SQL definition set and all of the data types defined in the definition set from SQL scripts, stored procedures, XQuery procedures, and XSLT procedures.

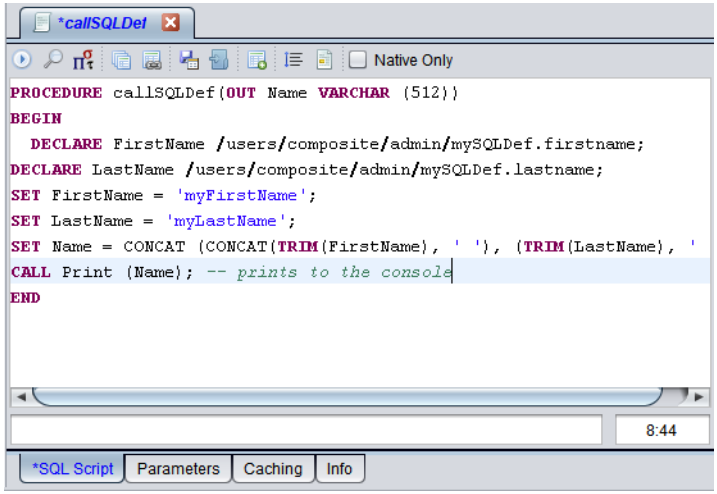
The procedure described here refers to the data types defined in a SQL definition set named mySQLDef:

Name	Base Type
firstname	CHAR(255)
lastname	CHAR(255)

To use an SQL-type definition set

1. Create a SQL script.
2. Reference the SQL definition set in the PROCEDURE using the TDV resource path and name.

   The SQL script below uses the SQL definition set named mySQLDef.



3. Execute the script to see the result.

Using an XML Definition Set

The procedure used in this section uses an XML definition set named XMLLib_Composite, which has the following XML schema and elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"

targetNamespace="http://www.compositesw.com/services/webservices/system/admin/resource"

xmlns:ns="http://www.compositesw.com/services/webservices/system/admin/resource"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
  >
<xs:simpleType name="Location">
<xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:complexType name="Address">
    <xs:sequence>
      <xs:element name="street" type="ns:Location"/>
      <xs:element name="number" type="xs:integer"/>
      <xs:element name="suite" type="xs:integer"/>
      <xs:element name="phone" type="xs:integer"/>
      <xs:element name="fax" type="xs:integer"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="CompositeAddress" type="ns:Address"/>
</xs:schema>
```

The procedure described here supplies values to the elements in the XML schema in the definition set XMLLib_Composite (Note: The type elements defined in the wsdltest definition set like TradePriceRequest and TradePrice are also referenced., page 389).

To use an XML-type definition set

1. Create a SQL script by right-clicking at an appropriate location in the resource tree and selecting New SQL Script, typing a name for it, and clicking OK.
2. Type the following script in the SQL Script panel of the script editor:

```
PROCEDURE ScriptComposite(OUT CompanyAddress
/shared/sources/definitionSets/XMLLib_Composite.
"{http://www.compositesw.com/services/webservices/system/admin/resource}Address")
    BEGIN
set CompanyAddress = '<ns:CompositeAddress
xmlns:ns="http://www.compositesw.com/services/
              webservices/system/admin/resource">
```

```
    <ns:number>2655</ns:number>
    <ns:street>Campus Drive</ns:street>
    <ns:suite>Suite 200</ns:suite>
    <ns:city>San Mateo</ns:city>
    <ns:zip>94403</ns:zip>
    <ns:phone>650-277-8200</ns:phone>
    <ns:fax>650-227-8199</ns:fax>
  </ns:CompositeAddress>';
    END
```

3. Execute the script to see the result.

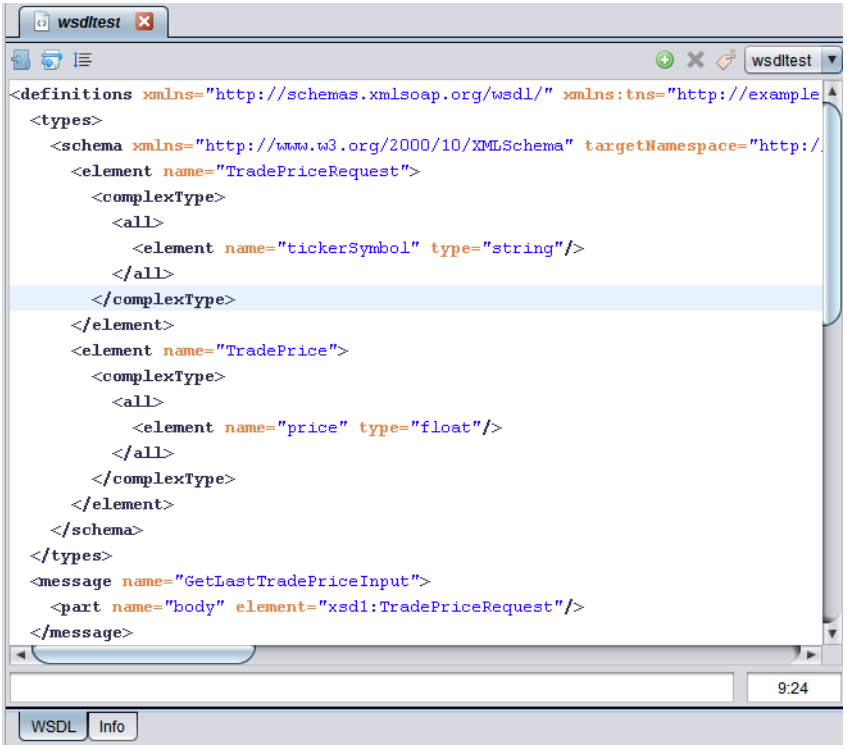
Using a WSDL Definition Set

You can use a WSDL definition set in a procedure or a view. The example here shows how a WSDL definition set can be used in a procedure.

To use a WSDL definition set in a procedure

1. In Studio, define and save a WSDL definition set.

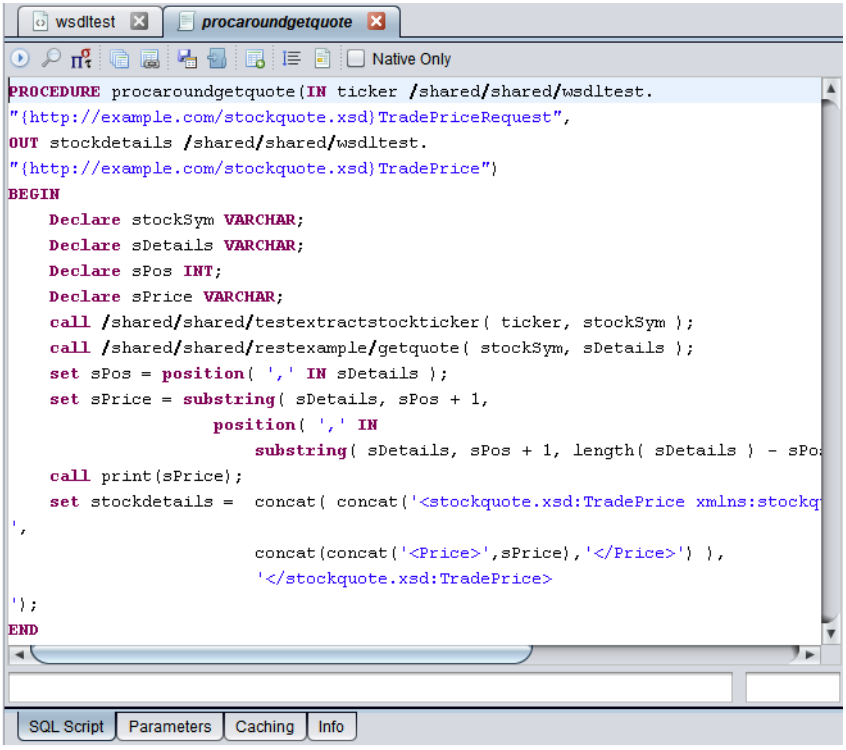
   This example uses a WSDL definition set named wsdltest.



2. Create or open a procedure.
3. Reference the WSDL definition set in the PROCEDURE using the TDV resource path and name.
4. The SQL script below uses the SQL definition set named mySQLDef.

   You can drag and drop the definition set resource from the Studio resource tree, causing the full name to appear in the procedure text panel. Refer the *TDV Reference Guide* for more details on the SQL Script parameters.

For example, if you reference the wsdltest WSDL definition set shown above, you would reference it in a procedure as shown in this example, where it is used for both input and output parameters.



Note: The type elements defined in the wsdltest definition set like TradePriceRequest and TradePrice are also referenced.

Triggers

This topic describes how to define triggers that cause an action to be performed under specified conditions.

About Triggers

A *trigger* is a resource that initiates a specific system activity to be performed when specified conditions occur. Triggers use a JMS event, a system event, a timer event, or a user-defined event to initiate specific system actions like:

- Execute a procedure
- Gather statistics
- Re-introspect data sources
- Send an email

Triggers can be scheduled to be performed periodically or can occur based on specified conditions. The triggering conditions can cause actions without requiring user interaction. When you create a trigger, it appears in the Studio resource tree.

Triggers provide the following benefits:

- One user can set multiple schedules (using separate triggers) on one resource.
- Many users with appropriate privileges can share the same schedule on a resource.
- Users can define their own events to trigger several actions asynchronously.
- Tasks and actions can be triggered in response to a system event, such as a request failure or the status of a data source.
- Procedures can be executed without sending notifications.

- Views and procedures can be executed and the results can be sent to one or more recipients.

Condition Types for a Trigger

This section describes the trigger condition types that Studio supports: JMS events, system events, timer events, and user-defined events.

- JMS Event—Enables the firing of an action based on the receipt of a JMS message on the specified queue or topic connection. The firing of the trigger is contingent on the match with the particular selector property specified using SQL condition syntax.
- System Event—Enables the firing of an action based on any system-generated event. User tasks can listen for such events. Two APIs—GetEnvironment and SetEnvironment—are available at */lib/util/*, and can be called at any time to listen for a system-generated event.

  When you specify System Event as the condition type for your trigger, the system provides a dynamic list of all available events for you to select from. The following System Events can be used for a trigger.

System event	When triggered or generated
CacheRefreshFailure	Triggered when a cache fails to refresh.
CacheRefreshSuccess	Triggered when a cache refresh is performed successfully.
ClusterServerConnected	Generated when a cluster node is connected.
ClusterServerDisconnected	Generated when a cluster node is disconnected.
ClusterServerShunned	Generated when a cluster node fails to send a heartbeat signal to the cluster timekeeper and has been shunned (temporarily excluded) from the cluster.

System event	When triggered or generated
DataSourceDown	Generated when a data source connection is lost. Data source connections are tested in any of the following cases: • Studio restart tests all data sources • New data source, configurations, and connections • Resource execution checks connection status • Committing/rolling back a transaction All reports of a data source being in DOWN status generate an event regardless of whether that data source was reported as down previously.
DataSourceUp	Generated when a data source connection is re-established. Data source connections are tested in any of the following cases: • Studio restart tests all data sources • New data source, configurations, and connections • Resource execution checks connection status • Committing/rolling back a transaction A Data Source Up event is generated only when the data source is first connected or when the status is changed from DOWN to UP.
ErrorsSpike	Generated when errors pass a predetermined number.
FailedLoginSpike	Any ten failed sign-in attempts within a ten minute period generate a FailedLoginSpike event.
MetricRestoreFailure	Generated when the metrics are trying to be restored, but they fail.
MetricsBackupFailure	Generated when the metrics backup fails.
MetricsPersistentFailure	Generated when metrics data fails to persist.
MetricsTruncationFailure	Generated when the truncation of metrics data fails.
RequestFailure	Generated when a request results in failure. Whenever any SOAP call or SQL execution is not accepted, produces an exception, or results in an error, this event is logged. This event is generated when closing any unsuccessful request.

System event	When triggered or generated
RequestInactive	Generated when a request is inactive. A request is considered inactive when it has no incoming or outgoing byte change beyond a configurable number of minutes. The value of the inactivity period is a server configuration parameter that can be configured using **Request Inactive Time**. By default there is no setting for the inactivity time.
RequestRunForTooLong	Generated when: • Enable All Events is set to true. • Request Run Time is enabled.
RequestsSpike	Generated when requests pass a predetermined number.
ResourceLock	Generated when resource locks are set on a resource so that only one person can modify the locked set.
ResourceUnlock	Generated when a resource is unlocked so that any user with sufficient rights and privileges can edit it.
ServerStart	Event is generated when TDV starts.
ServerStop	Event is generated when TDV stops.
StatisticsGatheringFailure	Event is generated when an attempt to gather statistics on a data source fails.
TransactionFailure	Event is generated on any transaction failure.

Making a trigger based on a system event does not enable a disabled event. All system events are enabled by default, but if they are disabled they require a manual server configuration change to enable them. For more information, see the *TDV Administration Guide*.

Triggers cannot enable a disabled data source.

- Timer Event—Lets you schedule periodic timed trigger events. Actions that can be timed to start are: executing a procedure, gathering relevant data from a data source for query processing, re-introspecting a data source, or sending email with the results of a procedure/view execution.

  To set a timer event, you define the start time, optionally specify a period or window of daily or periodic activity, and set the frequency of the trigger refresh to get the wanted periodicity. Triggers can be set to begin operation in the future and end after a predefined activity period.

You can optionally specify a recurrence restriction. If Recurrence Restriction is enabled, the trigger fires only on the selected days of the week and during the time window specified by the Start and End times, subject to the following rules:

  - If Start and End time are identical, no time restriction is applied.
  - If End time is greater than Start time, the trigger is restricted to fire only within the time window starting at Start time (inclusive) and ending at End time (inclusive).
  - If End time is less than Start time, the trigger is restricted to fire only within the time window starting at End time (inclusive) and ending at Start time on the next day (inclusive).

- User Defined Event—Utilizes a built-in procedure named GenerateEvent which is available at `/lib/util/`. This procedure can be called at any time to insert a user-defined event into the trigger system. You can use this functionality for letting an ongoing procedure call GenerateEvent to trigger a wanted action asynchronously.

See Condition Pane, page 671 for more information about the options for each condition type.

Action Types for a Trigger

Executing a procedure, gathering statistics on a data source, re-introspecting a data source, and sending email are the *trigger action types* that can be triggered for certain conditions.

- Execute Procedure—Lets you specify a procedure to be executed without sending any notification.
  - Exhaust output cursors—If the procedure is a SQL script that contains a PIPE cursor, and if the pipe needs to fully execute and load all rows to complete execution of a trigger (for example, to email a result set) checking this box forces the trigger to wait until all the rows are buffered and read before completing. If you leave this unchecked, the trigger returns while the pipe is buffering, cutting off the buffering of the pipe.
- Gather Statistics—Lets you specify a data source whose data statistics are to be gathered.
- Reintrospect Data Source—Lets you specify a data source to be re-introspected.
- Send email—Lets you specify a procedure or view to be executed, and send the results through email to one or more specified recipients.

See Action Pane, page 672 for more information about the options for each condition type.

Creating a JMS Event Trigger

These steps describe how to configure a JMS event trigger.

To create a JMS event trigger

1. If not already done, configure a JMS connector with a queue or topic connection factory for use with TDV. See the *TDV Administration Guide* for more information.
2. Right-click where you can add a new resource in the resource tree, and select New Trigger.
3. In the Input window, supply a name for the trigger, and click OK.
4. In the trigger editor that opens on the right, enable the trigger by selecting the Enable Trigger check box.
5. For Condition Type, select JMS Event.
6. In the Condition pane, enter values in these fields:

Field	Description
Connector	The name of your JMS connector (containing the configuration for initial context factory, JNDI Provider URL, and Queue Connection Factory) as it appears on the CONNECTOR MANAGEMENT page in Web Manager.
Destination	The name of the connection factory destination (queue or topic).
Selector	A JMS message selector allows a client to specify, by header field references and property references, the messages it is interested in. Only messages whose header and property values match the selector are delivered. For example, a selector condition using SQL WHERE clause syntax might look like: 'uname'='admin'

What it means for a message not to be delivered depends on the MessageConsumer being used.

— A message selector matches a message if the selector evaluates to true when the message's header field values and property values are substituted for their corresponding identifiers in the selector.

— A message selector is a String whose syntax is based on a subset of the SQL92 conditional expression syntax. If the value of a message selector is

an empty string, the value is treated as a null and indicates that there is no message selector for the message consumer.

7. Specify the action type and the options for the action as described in Action Types for a Trigger, page 395 and Action Pane, page 672.
8. On the trigger **Info** tab, specify the trigger properties. See Getting Information about a Resource, page 42 for all fields except Maximum Number In Queue which is described below and works different for JMS triggers:

   Maximum Number In Queue—For JMS triggers, this field is always set to 0 by the TDV server and you cannot change it. This is done to indicate that each JMS trigger can hold at most one message in memory at a time until it has been fully processed before receiving another pending message from the JMS server. That is, no message queuing is done inside TDV.
9. Save the trigger. If the Save option is not enabled, select File > Refresh All to refresh the resources and enable the Save option.

Creating a System Event Trigger

There are many system events that can be set as conditions for a trigger action. The APIs—GetEnvironment and SetEnvironment—that are available in /lib/util/ can be called at any time to listen for a system-generated event.

These steps describe how to create a sample procedure, create the system event trigger, and then test the trigger.

To create a sample procedure to call GetEnvironment

1. Within Studio, create a sample SQL script procedure, named CallsGetEnvironment as follows:

```
PROCEDURE CallsGetEnvironment()
    BEGIN
        DECLARE tpath VARCHAR(4096);
        DECLARE tname VARCHAR(4096);
        DECLARE ttype VARCHAR(4096);
        DECLARE tvalue VARCHAR(4096);
        DECLARE result VARCHAR(4096);
 CALL GetEnvironment('TRIGGER_PATH', tpath);
        CALL GetEnvironment('TRIGGER_EVENT_NAME', tname);
        CALL GetEnvironment('TRIGGER_EVENT_TYPE', ttype);
        CALL GetEnvironment('TRIGGER_EVENT_VALUE', tvalue);
        SET result = CASE
            WHEN (ttype IS NULL) THEN 'NULL'
            ELSE ttype
        END||' trigger '||CASE
            WHEN (tpath IS NULL) THEN 'NULL'
            ELSE tpath
```

```
    END||': '||CASE
        WHEN (tname IS NULL) THEN 'NULL'
        ELSE tname
    END||' = '||CASE
        WHEN (tvalue IS NULL) THEN 'NULL'
        ELSE tvalue
    END;
    CALL Log(result);
END
```

When this procedure is executed by a system event, the results are written out to the log.

To create a system event trigger

1. Right-click where you can add a new resource in the resource tree, and select New Trigger.
2. In the Input window, supply a name for the trigger, and click OK.
3. In the trigger editor that opens on the right, enable the trigger by selecting the Enable Trigger check box.
4. Select System Event from the drop-down list in the Condition Type section.
5. Select a system event from the System Event Name drop-down list on the right.
6. For Action Type, select the type of action to be triggered from the drop-down list.

Field	Description
Execute Procedure	To see how it works, specify the path to the procedure CallsGetEnvironment in the Action section. Select Exhaust output cursors, if the procedure is a SQL script that contains a PIPE cursor.
Gather Statistics	To see how it works, select the data source /shared/examples/ds_orders as the data source to be scanned.
Reintrospect Data Source	To see how it works, select the data source /shared/examples/ds_orders as the data source to be re-introspected.
Send E-mail	To see how it works, specify the path to the procedure CallsGetEnvironment in the Action section.

7. In the **Action** section, enter the options for the action type. The options are described in the section Action Types for a Trigger, page 395 and Action Pane, page 672.

For example, if you want email notifications sent, type and confirm the email addresses of the recipients to receive notification, as needed. You can specify a list of email addresses in the address fields by separating each one with a comma (,) or semicolon (;).

8. Click the Info tab.
9. Specify the Maximum Number In Queue field to the maximum number of times this trigger is to be fired if multiple periods have elapsed while a recurrence restriction prevents the trigger from firing. Most triggers should fire only once when the next available active operation window, but there might be cases when more than one trigger action should be initiated at the next opening of the recurrence restriction. This number should not be negative.
10. Save the trigger.

    If the Save option is not enabled, select File > Refresh All to refresh the resources and enable the Save option.

To test your system-event trigger

1. Select CacheRefreshFailure as the system event name in the Condition panel.
2. Select Send E-mail as the Action Type.
3. Specify the path to CacheRefreshFailure in the Resource Path field.
4. Refresh the cache of a broken view.

The specified mail recipients receive email notification and the result of running the path to procedure CallsGetEnvironment is written to the log.

Creating a Timer Event Trigger

This section describes how to create a timer event trigger.

To create a timer event trigger

1. Right-click in the location where you want to add a new trigger resource in the resource tree, and select New Trigger.

   Only valid trigger locations, like within My Home or Shared, offer the option to create a new trigger.
2. In the New Trigger window, supply a name for the trigger and click OK.

   The trigger editor opens on the right.

3. Enable the trigger by selecting the Enable Trigger check box.

   The Enable Trigger check box can be set to enable or disable the trigger at any time. Just save the trigger to set a changed value.

4. For Condition Type, select Timer Event.

5. In the Condition section, specify the schedule for the trigger. You set a timer event trigger to execute one time or repeat it at intervals based on minutes, hours, days, weeks, months, or years and specify the recurrence as follows:

   a. To schedule the event to be executed one time, select Exactly Once and skip directly to the Start on section to schedule when the trigger is to be fired.

   b. To schedule a periodic event, select Periodic, specify the refresh frequency (Refresh every), and select the time unit from the drop-down list.

   c. To schedule a periodic event, also specify the date and time for the first execution using the Start on (date) and at (time) fields.

   The date and time entered specify the moment of the first triggering event. For example, if a daily event is set for 11:55 AM three days in the future, it runs at 11:55 AM in three days and then every day at the same time thereafter.

   The date and time also mark the beginning of the Refresh every period. For example, if you specify Refresh every 1 hour, and you specify 9:35 AM in the Starting at section, a firing condition occurs at thirty-five minutes past the hour every hour from 9:35 AM onward.

   If any recurrence restriction is in effect), the trigger can only *fire and initiate an action* within the time range and on the selected day(s). The actual firing of an action is independent of the period you specify; that is, queuing a firing does not change when the next firing condition will be met.

   d. Optionally, check Enable for the Recurrence Restriction (Optional) and check days of the week and a time range to restrict trigger operation to a time window on specified days of the week.

   By default, no recurrence restriction is in effect, and so a scheduled trigger fires whenever the specified period has elapsed. No recurrence restriction is in effect if every day of the week is selected and the start and end time are the same.

   The recurrence restriction window begins a period during which a trigger fires and initiates the specified action. If the trigger conditions are met outside of the restricted window, then the trigger queues a firing event for the next recurrence restriction start time. The actual number of firing events allowed in the queue is configurable. See Getting Information about a Resource, page 42.

If the trigger should be restricted to firing within a specific window of operation:

— Specify the days on which the trigger can fire.

— Specify the **Start** and **End** time when the trigger can fire.

With Enable checked, the trigger fires only on the selected days of the week and during the time window specified by the Start and End times, subject to the following rules:

— If Start and End time are identical, no time restriction is applied.

— If End time is greater than Start time, the trigger is restricted to fire only within the time window starting at Start time (inclusive) and ending at End time (inclusive).

— If End time is less than Start time, the trigger is restricted to fire only within the time window starting at End time (inclusive) and ending at Start time of the next day (inclusive).

More complex scheduling can be created by combining the effects of multiple triggers that can call the same actions with different scheduling parameters.

Note: If you check Enable in the Recurrence Restriction (Optional) section but you do not check any day of the week, the trigger does not have any valid firing period and is effectively disabled regardless of other settings.

e. If the TDV Server is to be a member of a clustered group of servers, then check. Only once per cluster to avoid multiple actions being triggered simultaneously from every TDV Server instance.

6. For Action Type, select the type of action to be triggered from the drop-down list.

7. In the Action section, enter the options for the action type. The options are described in the section Action Pane, page 672.

   Select Exhaust output cursors, if the procedure is a SQL script that contains a PIPE cursor.

   For example, if you want email notifications sent, type and confirm the email addresses of the recipients to receive notification, as needed. You can specify a list of email addresses in the address fields by separating each one with a comma (,) or semicolon (;).

8. Click the Info tab. In the Maximum Number In Queue field in the Queue Properties section, specify the maximum number of trigger firings to queue if multiple periods have elapsed while a recurrence restriction prevents the trigger from firing.

Most triggers should fire only once within the next available active operation window, but there might be cases when more than one trigger action should be initiated at the next opening of the recurrence restriction. This number should not be negative.

9. Save the trigger.

If the Save option is not enabled, select File > Refresh All to refresh the resources and enable the Save option.

Creating a User-Defined Event Trigger

When creating a user-defined event trigger, you first need to create a user-defined event that can function as a condition for a trigger action. Then you create the user-defined event trigger. Finally, you test the trigger.

Triggers are created like other TDV resources, but they cannot be published.

User-defined events are generated when the built-in procedure GenerateEvent (at /lib/util/) is called with an event name and value as its arguments.

User-defined event triggers execute once per cluster. Timer-based triggers can be designated as being cluster-aware and they fire on all cluster nodes on which the user-defined event is generated.

To create a user-defined event

1. Create a SQL script procedure named CallsGenerateEvent to call GenerateEvent, as in the following example:

```
PROCEDURE CallsGenerateEvent()
    BEGIN
        CALL GenerateEvent('runAReport', ' ');
    END
```

This procedure calls GenerateEvent, which in turn creates a custom event with the name runAReport. Whenever this procedure is executed, GenerateEvent inserts the custom event named runAReport into the trigger system. The custom event is available for you to specify as a User Defined Event Name in the Condition section of a user-defined event trigger.

To create a user-defined event trigger

1. Right-click where you can add a new resource in the resource tree, and select New Trigger.
2. In the Input window, supply a name for the trigger, and click OK.

3. In the trigger editor that opens on the right, enable the trigger by selecting the Enable Trigger check box.
4. Select User Defined Event from the drop-down list in the Condition Type section.
5. In the User Defined Event Name field, supply a name for the event.

   You can use a regular expression, for example, run*, to match more than one similar user-defined event name. For this example, you would supply the name runAReport.
6. For Action Type, select the type of action to be triggered from the drop-down list.
7. In the Action section, enter the options for the action type. The options are described in the section Action Pane, page 672.

   Select Exhaust output cursors, if the procedure is a SQL script that contains a PIPE cursor.

   For example, if you want email notifications sent, type and confirm the email addresses of the recipients to receive notification, as needed. You can specify a list of email addresses in the address fields by separating each one with a comma (,) or semicolon (;).Click the Info tab. Specify the Maximum Number In Queue field to the maximum number of times this trigger is to be fired if multiple periods have elapsed while a recurrence restriction prevents the trigger from firing. Most triggers should fire only once when the next available active operation window, but there might be cases when more than one trigger action should be initiated at the next opening of the recurrence restriction. This number should not be negative.
8. Save the trigger.

   If the Save option is not enabled, select File > Refresh All to refresh the resources and enable the Save option.

To test your user-defined event trigger

- Run the procedure CallsGenerateEvent that was created as described in Creating a User-Defined Event Trigger, page 402.

Creating Email Alerts

You can trigger email notifications for TDV actions or events.

Tip from an expert: The following configuration parameters are left over from functionality that does not send email alerts: Email Addresses for CC, Enable Email Events, Email Addresses.

To enable email alerts

1. Open and log in to Studio.
2. From the Administration menu, choose Configuration.
3. Navigate to Server > Configuration > E-Mail.
4. Set values for the following:

Configuration Parameter	Description of Value	Example
From Address	Email address that you want to appear in the From line for alerts.	meg@queenbeesknees.net
SMTP Host Name	Name of the email server host.	javamail.queenbeesknees.com
Maximum number of rows included in email attachment.	If set to 0, there is no restriction on the size of the email attachment. If set to a value greater than 0, the value is used as the maximum number of rows allowed for the attachment.	0

5. Save and exit the Configuration window.
6. From the Studio resource tree, right-click and select New Trigger.
7. Name and enable it.
8. Set the type of event that you want to trigger the alert. For example, a system event such as a cache refresh or data source going down. For information on how to set the different types of triggers, see the *TDV User Guide*.

Choose System Event to collect information for typical TDV events including, Metrics collection, caching actions, and request spikes.

Typical Event Areas	System Event Name
Metrics Alerts	MetricsPersistentFailure
	MetricsTruncationFailure
	MetricsBackupFailure
	MetricsRestoreFailure
	StatisticsGatheringFailure
Caching	CacheRefreshFailure
	CacheRefreshSuccess
Cluster Management	ClusterServerJoined
	ClusterServerConnected
	ClusterServerDisconnected
	ClusterServerShunned
Data Source Management	DataSourceDown
	DataSourceUp
Request Management	RequestFailure
	RequestInactive
	RequestRunForTooLong
	RequestsSpike
	TransactionFailure
Resource Management	ResourceLock
	ResourceUnlock
Errors and Login Management	ErrorsSpike
	FailedLoginSpike
Server Management	ServerStart
	ServerStop

Typical Event Areas	System Event Name
Trigger Management	TriggerStart
	TriggerEnd
	TriggerFail

9. Select the Action Type of Send E-mail.
10. Specify a Resource path. For example, /shared/examples/ds_orders/tutorial/customers.
11. Type the email addresses for which to send the email alerts. For example, meg@queenbeesknees.net.
12. Type a meaningful Message Subject.
13. Type a meaningful Message Body.
14. Save the trigger.

Publishing Resources

This topic describes publishing of TDV-defined resources. You can publish almost any object that you create in TDV, including views and procedures.

Updating Published Resources

Published resources are automatically updated if you change their source resource using Studio. Typically you should only add new resources to your published folders.

About Publishing Resources

Resources are initially defined in a Studio user workspace or in a Studio shared workspace, but they are not available for end-user consumption until they are published. You can publish any type of tabular data (views, data stored in a table in a text file, relational database tables, or LDAP tables) or procedure (SQL scripts, Java procedures, transformations, parameterized queries, and packaged queries). You can also publish an entire data source. The structure of the data source is preserved in the published version.

TDV does not recommend publishing a database table directly. The recommendation is to create a composite view of the table and publish the view.

Any time an original resource is modified, the corresponding published resource reflects the change. Also, you can access and use the data from an original resource through the corresponding published resource.

You can make resources available to client applications by publishing them as Data Services. For quick steps, see Publishing Resources (Creation, Access, and Deletion), page 414.

Resources can be published as:

- Databases—You can create TDV databases, catalogs, and schemas in this container to publish tabular data and procedures. You can create as many TDV databases and schemas as you need. You can query the published tabular data as you would query any normal database.

  For details, see Publishing Resources to a Database Service, page 416.

  Optionally for database services, you can learn about the following options to decide if you would like to implement them in your environment:

  — About the TDV DDL Feature, page 409

  — About the OData Feature, page 410

- Web services—You can create Web services within this container. Within a Web service, you can publish tables and procedures as links to the actual resources.

  For details, see Publishing Resources to a Web Service, page 421.

  Optionally for Web services, you can decide how to manage XML objects for implementation in your environment:

  — Accessing REST-Based Web Services Using JSON, page 414

Resources are published to the Data Services node in the resource tree. The Data Services node is a published interface that should be organized for the convenience of the client that needs to access that data.

For details on how client applications can query and consume published resources, see the *TDV Client Interfaces Guide*. Connectors must be defined prior to publishing resources over JMS. See "Configuring TDV Data Connections" in the *TDV Administration Guide*.

About the TDV DDL Feature

In addition to publishing to a database data service, you can use the TDV DDL (Data Definition Language) feature to manipulate tables in a physical data source. Also included in this topic are:

- Managing Data Type Mappings for the TDV DDL Feature, page 409
- TDV DDL Limitations, page 410
- Netezza DDL Creation Limitations, page 410

When your client application creates a table in a data source, it uses the connections and protocols established with TDV. A created table is immediately added to the TDV resource tree and can be manipulated like any other table object within TDV. TDV DDL CREATE TABLE supports the following features (as supported by the underlying data source):

- Column name specification
- Column type specification (after consulting the capabilities framework to determine type mapping appropriate to the data source)
- Column nullability specification
- Primary key specification

If your client application needs to DROP a table, the table is dropped from both the data source and the TDV resource tree directly. No re-introspection of the data source is needed to see the change within TDV.

This feature can be used to improve the performance of queries or reports, because it lets you create and drop temporary tables as needed to optimize actions.

Managing Data Type Mappings for the TDV DDL Feature

TDV can create tables in a wide variety of data source types, and so TDV consults its capabilities framework to determine the mappings between the data type name in the TDV query and the corresponding data type in the native data source. You can override the data type mappings, or specify new mappings, by defining custom data adapters for TDV. For more information, see Adding and Removing Data Source Resources, page 74.

Based on testing, we recommend that you use the following data type mappings for custom adapters that you intend to use with MicroStrategy.

Database	Custom Data Type Mapping
MySQL with MicroStrategy	FLOAT -> DOUBLE
Oracle with MicroStrategy	DOUBLE -> NUMERIC

TDV DDL Limitations

- If client applications are open and accessing data, and the temp table containers change, the updated settings might not be visible until the client applications sign in again. Temp tables created before the change are not deleted from the database.
- For data sources that make use of pass trough login, active temp tables might not get cleaned up during a TDV Server shutdown. TDV logs all those tables which it cannot clean up in the cs_server.log file at server startup.

Netezza DDL Creation Limitations

TDV cannot use DDL to create the following data types in Netezza data source tables:

• BINARY	• BLOB	• CLOB	• IMAGE	• LONG	• LONGRAW
• NCLOB	• NTEXT	• TEXT	• TIMESTAMP	• VARBINARY	

About the OData Feature

OData is a Web protocol that provides JDBC-like access for querying and updating data from a variety of sources, including relational databases, file systems, content management systems and websites. OData allows you to query a data source through the HTTP protocol. Optionally, you can configure transport-level security and authentication methods. TDV uses odata4j to parse OData queries.

TDV supports the reformatting of results, and several system query options. These standard OData syntax options are discussed on the OData website, odata.org.

- OData Support Limitations, page 411

OData Support Limitations

TDV does not support the following features:

- Procedures and resources where a primary key is *not* defined
- OData import functions
- OData navigation links
- OData is-or-is-child-of (IsOf) functions
- $count
- The resource you want to publish through OData, must have a primary key defined. For details, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.

Reformatted Results

By default in TDV, all resources that are published as database data services are available through OData in the XML-based Atom format. Client interfaces (the applications requesting data) can tell OData to reformat the Atom results to JSON using $format=json (instead of the default $format=atom) in the client interface definitions. For example:

```
http://services.odata.org/OData/OData.svc/ds_orders?$format=json
```

A client that wants only JSON responses might set the header to something similar to this:

```
GET /OData/OData.svc/ds_orders HTTP/1.1 host: services.odata.org
accept: application/json
```

Client applications can choose to support only specific content types, or to support all content types. To accept all content types, the header might look like this:

```
GET /OData/OData.svc/ds_orders HTTP/1.1 host: services.odata.org
```

System Query Options

You can use options to filter out unqualified records on the OData server side, before the data is returned. This can significantly reduce the amount of data transferred, and improve performance, compared to applying filters using the SQL WHERE clause on the TDV side.

Expand Option

The expand option retrieves records and includes their associated objects. For example,
`http://services.odata.org/OData/OData.svc/ds_orders/?$expand=orders`

For each entity within the product's entity set, the values of all associated orders are represented in-line.

The corresponding TDV SQL query would be:

```
SELECT * from
/shared/OData/ODataDemo/DemoService/Products('?$expand=orders')
```

Filter Option

The filter option retrieves only records that meet certain criteria. For example,
`http://services.odata.org/OData/OData.svc/ds_orders/?$filter=orderid gt 1000`

This request is used for query and return only records whose orderid value is greater than 1000.

The corresponding TDV SQL query would be:

```
SELECT * from
/shared/OData/ODataDemo/DemoService/Products('$filter=orderid gt
1000')
```

OrderBy Option

The orderby option arranges the returned records in the order of a designated entity's values. For example,
`http://services.odata.org/OData/OData.svc/ds_orders/?$orderby=customerid`

This request displays records in the order of customerid values.

The corresponding TDV SQL query would be:

```
SELECT * from
/shared/OData/ODataDemo/DemoService/Products('$orderby=customerid'
)
```

Skip Option

The skip option retrieves records by skipping a specific number of rows. For example,
`http://services.odata.org/OData/OData.svc/ds_orders/?$skip=2`

This request returns all records except the first two rows.

The corresponding TDV SQL query would be:

```
SELECT * from /shared/OData/ODataDemo/OData.svc/ds_orders('$
skip=2')
```

Top Option

The top option retrieves the first *n* records. For example,

```
http://services.odata.org/OData/OData.svc/ds_orders/?$top=4
```

This request returns the first four rows.

The corresponding TDV SQL query would be:

```
SELECT * from /shared/OData/ODataDemo/DemoService/Products('$
top=4')
```

Select Option

The select option retrieves values in the column with the specified name. For example,

```
http://services.odata.org/OData/OData.svc/ds_orders/?$select=shipn
ame
```

All returned records display only the values in the column named shipname. The other columns display [NULL].

The corresponding TDV SQL query would be:

```
SELECT * from /shared/OData/ODataDemo/DemoService/Products('
?$select=shipname')
```

Combinations of Options

You can concatenate system query options by putting ampersands between them. For example,

```
http://services.odata.org/OData/OData.svc/ds_orders/?$filter=order
id gt 1000&$top=2
```

This request return the first two rows whose orderid is greater than 1000.

The corresponding TDV SQL query would be:

```
SELECT * from /shared/OData/ODataDemo/DemoService/Products('
?$filter=orderid gt 1000&$top=2 ')
```

Accessing REST-Based Web Services Using JSON

JSON data objects, including array objects, would need to be created and any nested array objects need to be present before the results can be transferred. Arrays in JSON require that the data be held in memory, which for extremely large arrays can cause out of memory errors. If there are no arrays, then TDV will stream the data for better performance.

Typically, resources that contain XML data types might involve hierarchical data resulting in JSON arrays. So, you should avoid consuming the data from these resources through Web services using REST with JSON format if the resource you are consuming contains one of the following:

- The type of output parameter, or the column type in the cursor binding in one of output parameters is an XML type that might result in JSON array if the amount of data is large enough to require an array.
- Extremely large amounts of data from an XML formatted object.

For example, the following procedure is likely to perform better if it is consumed through an XML format rather than a JSON format:

```
proc (out MyFavFunct XML)
```

Review the views and procedures that you want to publish to determine if they contain XML types that will result in JSON arrays or cursors with XML data types. Depending on what you find, determine the best way for you to consume your resources.

See Publishing Resources to a Web Service, page 421 for more information about consuming resources using JSON.

Publishing Resources (Creation, Access, and Deletion)

When selecting names for your data services and resources, consider that special characters can make it difficult or impossible for clients (including Business Directory or Deployment Manager) to display or consume the published resources. For example, Business Directory cannot see the data for a data source with the name “dsInv/publish”.

You can do the following for published resources:

- Access its underlying source as it exists in Studio.
- View the resource content if it is a table or view.
- Rebind to a difference source. See Rebinding a View, page 248 for details.

To publish resources

1. From the resource tree, select one or more resources that you want to publish.
2. Right-click and select Publish.
3. Select from the list of existing data services or create a new data service:
   a. Select Databases or Web Services.
   b. Click Add Composite Service or Add Composite Web Service. Depending on what is selected, the buttons are active or unavailable.
   c. Type a name for the new data service that you want to use to hold your published resource.
   d. Click OK. The new data service is created.
   e. Click OK to save the resource into the new data service.
4. Finish definition of your data service by using the instructions in one of the following sections:
   — Publishing Resources to a Database Service, page 416
   — Publishing Resources to a Web Service, page 421

To access the underlying source of a published resource

1. Expand the published object so that you can see the resources it contains.
2. Double-click the published resource to open it.
3. In the editor that opens on the right, click to open the Info tab.
4. Click Open Source.

To view the contents of a published table or view resource

1. Double-click the published table or view to open it.
   — Click Show Contents, or right-click at the table or view's name in the resource tree and select Show Contents.

To delete published resources

- Right click the parent folder or single resources and select Delete.

Publishing Resources to a Database Service

To make tabular data and procedures available for querying and manipulation as a TDV database service, you need to publish them to a container (folder) under Data Services/Databases.

You can create as many TDV databases and schemas as you need, and organize them in multiple folders. The published tabular data can be queried like any normal database.

This section describes general publishing instructions. If you would like to define one of the optional TDV DDL or OData features for your database service, see the following sections:

- Publishing Resources to a Database Service with OData, page 417
- Configuring DDL for a Data Service, page 419

Considerations

If you plan to consume your published data from an Excel spreadsheet, you must publish the data tables using catalogs. For example, create a catalog under Data Service/Databases/<catalog> so that the data from the resources there can be used through an Excel client interface.

When selecting names for your data services and resources, consider that special characters can make it difficult or impossible for clients (including Business Directory or Deployment Manager) to display or consume the published resources. For example, Business Directory cannot see the data for a data source with the name “dsInv/publish”.

To publish a resource to a database service

1. If necessary, create a folder under Data Services/Databases where you want to publish the resource.
2. Highlight one or more tables, view, or procedure in the resource tree.
3. Select one or more resources, right-click, and select Publish or select Resource > Publish from the Studio menu bar.

   The Publish <resource_path_and_name> window appears.
4. Select the folder under Data Services/Databases where you want to publish the resource.

   The resource name is displayed in the Name field at the bottom of the dialog.

5. Optionally, type a different name in the Name field—for example, CompositeView_Published_As_a_DB.
6. Optionally, check Overwrite existing published resources of the same name.
7. Click OK.

   The resource now appears in the resource tree in the folder where you published it.

Publishing Resources to a Database Service with OData

You can use OData to publish resources to a database service.

You can use multiple columns for filtering a published OData resource by creating a unique index for the resource and then selecting two or more columns as index columns.

To publish resources for a database service and define OData options

1. Follow the instructions in Publishing Resources to a Database Service, page 416.
2. Open the database service that you created (at the level right below Data Services/Databases).

   The database service opens on the OData panel.

   Views or tables published through OData require a primary key. If a primary key does not exist, you must create one. For details, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.

   The list of endpoint Service URLs of the OData producers is exposed. There is one producer endpoint for each data source, catalog, or schema in the published database that contains table links.
3. In the Transport Level Security section of the OData panel, select from the following options.

TLS Option	Description
Enable SSL/TLS	Allows access to OData endpoint through the HTTPS protocol with SSL/TLS to secure data over the network. You must check this option to access either of the next two options. This option is checked and dimmed (unavailable to clear) after Require SSL/TLS and/or Validate Client Certificate is checked.

TLS Option	Description
Require SSL/TLS	When this is checked, HTTP access is disabled, and access to OData endpoints must use HTTPS.
Validate Client Certificate	When this is checked, the client must present a certificate to the server during authentication, and the server must validate it.

4. In the HTTP Authentication Methods section, select one or more of the options listed in the table.

Authentication Options	Description
Basic	Allows users to present plain-text username and password credentials for authentication.
NTLM	Enables NTLM protocol for authentication of username and password credentials.
Digest	Enables a digest-based protocol for authentication of username and password credentials.
Negotiate	Enables client and server to negotiate whether to use NTLM or Kerberos Protocol (thus making Kerberos available).

5. In the Unsupported section, review the list of resources that are not available through OData.

   The Reason column explains why resources are unsupported. Determine if you want to fix any issues that caused them to appear on this list. For example, you could add primary keys to resources that do not have them (Defining Primary Key for a View or Table in the Indexes Panel, page 238).

6. Save your changes.
7. Optionally to use multiple columns for filtering, specify one or more columns for filtering, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.

Configuring DDL for a Data Service

The TDV DDL feature allows TDV to use DDL to directly create or drop tables in a physical data source. On the TDV DDL tab you can specify the path to a relational container (catalog or schema) for DDL submission. The DDL will be submitted by TDV clients (that is by a client that uses JDBC, ODBC, or ADO.net protocols).

Having multiple locations within a data source for the manipulation of data, helps avoid conflicts when several users are attempting to access the same tables. This allows temp table creation in one or more data sources where related query work is being performed. In this way you can take full advantage of the TDV optimizer's capability to push an entire query down to a database when all joined objects reside in a common data source. Data sources have a variety of proprietary optimizations, so by being able to determine where a query will run, you can determine the most optimal data source for your query to run.

For information on Netezza DDL creation limitations, see About the TDV DDL Feature, page 409.

Data services support the following DDL statements:

- CREATE TABLE AS SELECT
- DROP TABLE

DDL Studio and User Security Requirements

- The published catalog or schema and the target data source container must already exist and you must have read/write access to the objects to perform the specification creation.
- If the catalog or schema exists, you must have read/write access to the objects to perform the specification creation.
- You must have access to the underlying data source and database container using pass-through or non-pass-through authentication.

To specify the path to a relational container for DDL submission

1. Follow the instructions in Publishing Resources to a Database Service, page 416.
2. Make sure that the data source where the DDL statements will be run has all the necessary user permission settings so that the TDV and client users will have the ability to execute the DDL statements.
3. Open the database service.

4. Select the Composite DDL tab.
5. Type or browse to the container path where a client might later want to CREATE or DROP tables.

   You can create the temporary tables in Oracle, Netezza, SQL Server, Teradata, MySQL, and DB2. Support for creation of other TEMP tables depends on the physical data source where you want to create them.
6. Optionally, if your data source supports it, you can specify more than one location where tables can be created or dropped. These locations are typically called Catalogs or Schemas.
   a. Type or browse to the Published Container Path.
   b. Type or browse to the Physical Container Path (as it exists in Studio).
   c. Add more locations as necessary.
7. Save your changes.

See the *TDV Client Interfaces Guide* for an example of how to configure ODBC and MicroStrategy.

Configuring TEMP Table Space for a Data Service

Temporary tables come in handy for a variety of business integration tools. The temporary tables allow the tools to store filters for visualizations because of their simplicity. Temporary tables also create space to accommodate DDL capabilities that you might also configure for your data service. Specifically, the temporary tables can streamline the creation and removal of the table for use during a working session.

You can use Studio to configure a location where you would prefer to have temporary tables created for your data services.

You can create the temporary tables in Oracle, Netezza, SQL Server, Teradata, MySQL, and DB2. Support for creation of other TEMP tables depends on the physical data source where you want to create them.

For information on limitations, see About the TDV DDL Feature, page 409.

To specify the temporary table locations

1. Follow the instructions in Publishing Resources to a Database Service, page 416.
2. Open the database service.

3. Select the Temp Tables tab.
4. Type or browse to the container path where a client might later want to temporary tables.
5. Optionally, if your data source supports it, you can specify more than one location where tables can be created or dropped. These locations are typically called Catalogs or Schemas.
   a. Type or browse to the Published Container Path.
   b. Type or browse to the Physical Container Path (as it exists in Studio).
   c. Add more locations as necessary.
6. Save your changes.

Publishing Resources to a Web Service

To make tabular data and procedures available for querying and manipulation as a TDV Web service, you need to publish them to a container (folder) under Data Services/Web Services.

You can create as many TDV databases and schemas as you need, and organize them in multiple folders.

This section describes how to publish views and procedures as WSDL (Web Services Definition Language) services. The following topics are covered:

- About WSDL and Web Service Data Services, page 421
- Publishing a SOAP Data Service, page 422
- Publishing a Contract-First WSDL and SOAP Data Service, page 428
- Moving the Namespace Declaration From Elements to the Soap Header, page 431
- Publishing a REST Service, page 432

About WSDL and Web Service Data Services

Web Services Description Language is an XML-based language that describes a Web service. The WSDL is a contract that explains the details of how that service works. A WSDL file specifies how clients can submit input to an operation, and what kind of output clients can expect in return.

Web services are Web-based applications that dynamically interact with other Web applications using an XML message protocol such as SOAP or REST. Web services can be bound to any message format and protocol. The binding profile allows specification of the HTTP transport protocol, literal encoding scheme, and document message style.

Considerations

For WSDL data services use explicit cursor definition. The cursor should name the column.

When selecting names for your data services and resources, consider that special characters can make it difficult or impossible for clients (including Business Directory or Deployment Manager) to display or consume the published resources. For example, Business Directory cannot see the data for a data source with the name "dsInv/publish".

Publishing a SOAP Data Service

This section describes how to add a SOAP 1.1 or 1.2 compliant data service.

To publish a SOAP data service

1. Select the Web Services node under the Data Services node in the resource tree.
2. Right-click and select New Composite Web Service.
3. In the Add Composite Web Service dialog, type a data service name for your new Web service.
4. Click OK.
5. From the resource tree, open the Web service that you just made.
6. Optionally, publish tables, views and procedures to the Web service.
    a. Select one or more tables, view, or procedure in the resource tree and right-click.
    b. Select Publish.
    c. Optionally, in the Publish window, edit the name to use for the published resource.
    d. Select the Web service to which you want the resource published.
    e. Click OK.

If, for example, you published CompositeView to the Web service you just created, and the Web service is still opened to the SOAP panel, it would be updated to display the Web service properties, operations, and parameters.

7. In the Service portion of the screen, type or select values for the following properties.

   If a property *requires* a value, a default value is already assigned to it.

Properties	Required	Description
Enabled	Yes	True (the default value) tells the system you want to use the information that you have defined on this tab.
Target Namespace	Yes	Replaces the default value with the location of your SOAP installation.
Port Type Name	Yes	Name of the port. SOAP Web services are described with WSDL. WSDL defines a Port Type as an abstract collection of operations. Use a WSDL to describe the Web service and create a Port Type entry in the WSDL. The port type name is arbitrary, though it must be a legal QName.
Binding Name	Yes	A unique binding name that can be referenced from elsewhere in the WSDL definition.
Enable Contract First	Yes	The default value is false. Select true to point to your predefined WSDL and WSDL operations. For information on how to create the service, see Publishing a Contract-First WSDL and SOAP Data Service, page 428.
Enable MTOM	Yes	Valid values are true or false. Set to true to enable the use of Message Transmission Optimization Mechanism (MTOM), a method of sending binary data to and from Web services. After it is enabled you must complete the definition of binary parameter types.
Security Policy	Yes	List shows all of the available policies that are defined under localhost/policy/security.
SAML Validator Class	No	Name of the Java class that lets users validate SAML assertions.

Properties	Required	Description
Subject Mapper Class	No	Define a Java class for your SAML content using TDV's extended Java API Subject Mapper, which can be found in: <TDV_install_dir>\apps\extension\docs\com\compositesw\extension\security\SubjectMapper.html Use method samlSubjectToCompositeUser to map samlSubject to an existing TDV user, so that this SAML principal has the same privileges as that user. After you create the Java class, save the JAR file to the <TDV_install_dir>\apps\server\lib folder.
Endpoint URL Path	Yes	The path of the endpoint URL for this Web service. Default is the name (with spaces removed) you used for the Web service.
SOAP11 Context Url Path		The path in which a SOAP object should appear. Associating an object to a certain page on your site.
SOAP12 Context Url Path		The path in which a SOAP object should appear. Associating an object to a certain page on your site.
WSDL URLs		Property heading.
HTTP/SOAP 1.1		The URL of the HTTP SOAP 1.1 WSDL file that can be used to access the operation.
HTTP/SOAP 1.2		The URL of the HTTP SOAP 1.2 WSDL file that can be used to access the operation.
HTTPS/SOAP 1.1		The URL of the HTTPS SOAP 1.1 WSDL file that can be used to access the operation.
HTTPS/SOAP 1.2		The URL of the HTTPS SOAP 1.2 WSDL file that can be used to access the operation.

If you have published views or procedures to this service, you can perform following steps.

8. From the upper part of the Operations section, select a Web service table, view, or procedure that is available for use.

9. In the lower part of the Operations section, type or select values for the properties listed in the following table.

Property		Description
Security Policy		Lists all available policies defined under localhost/policy/security.
SOAP Action		Typically, this field is unused. If this value is defined as part of your contract-first WSDL, the value is displayed in this field.
Input Message		
	Parameter Style	This required value applies to all parameters. • Select WRAPPED when multiple parameters might use the BODY location. The wrapper element is the child of the SOAP BODY and the parameter elements are children of the wrapper element. • Select BARE when exactly one parameter with a location of BODY and its element is the only child of the SOAP BODY. All other parameters of the same direction must be mapped to another location—for example, HEADER.
	Wrapper	
	Element Name	Unique name to give to the wrapper element. For example, if you type wrappedOrderParams here, the WSDL XML includes an element: <wrappedOrderParams> </wrappedOrderParams>
	Element Type	Row element type.
	Message Name	Unique name for the wrapper message.
	Part Name	Name of the message part that represents the wrapper element.
	Security Policy	Security policy to use on the input message.

Property		Description
Output Message		
	Parameter Style	This required value applies to all parameters. • Select WRAPPED when there might be multiple parameters that use the BODY location. The wrapper element is the child of the SOAP BODY and the parameter elements are children of the wrapper element. • Select BARE when there is exactly one parameter with a location of BODY and its element is the only child of the SOAP BODY. All other parameters of the same direction must be mapped to another location, for example, HEADER.
	Wrapper	
	Element Name	Unique name to give to the wrapper element. For example, if you type wrappedOrderParams here, the WSDL XML includes an element: <wrappedOrderParams> xxx </wrappedOrderParams>
	Element Type	Element type of your row element.
	Message Name	Unique name to give to the wrapper message.
	Part Name	Name of the message part that represents the wrapper element.

10. Repeat for each operation or view you want to define for the Web service.
11. If your operation has parameters, go to the upper part of the Parameters portion of the screen and select a parameter from the list.
12. In the lower part of the Parameters portion of the screen, you can view and edit the properties and values of the parameter you selected in the upper part of the Parameters section.

Property	Description
Name	Name of the parameter. Cannot be edited.
Element Name	Fully qualified name to give to the input parameter, output parameter, or output cursor; or name (relative to the cursor path) to give to the column.

Property	Description
Direction	(Not listed for columns) Indicates the direction of the cursor. Cannot be edited.
Binding Location	When the selected operation is a procedure, you can define binding locations. If the message is BARE, exactly one parameter can use the BODY binding location. The available locations for input are: BODY, HEADER The available locations for output are: BODY, FAULT, HEADER, HEADERFAULT
Row Element Name	(Cursor only) Unique name to give to the row element of the cursor. For example, if you type myFavoriteRow here, the WSDL XML includes an element: <myFavoriteRow> </myFavoriteRow>
Row Type Name	(Cursor only) Type value of this row element.
Cursor Type Name	(Cursor only) Unique name for the type that defines the global cursor element.
MTOM CONTENT TYPE	If the parameter is binary, you should set the parameter values to VALUE_TYPE_BINARY, VALUE_TYPE_BLOB, or VALUE_TYPE_CLOB. The content-type values are available from a list of values. The default is application/octet-stream (a binary file). This header indicates the Internet media type of the message content, consisting of a type and subtype, that is accepted.

13. Repeat for each parameter that is to be defined for the Web service.
14. Save your definition.

Publishing a Contract-First WSDL and SOAP Data Service

You can make existing WSDL definitions available through TDV by publishing them as a Web service that uses SOAP 1.1 or 1.2. When publishing contract-first WSDL, you start with the WSDL contract, and use TDV to fill out the rest of the contract. In this situation, use TDV to import existing WSDL files by using a definition set. Start with the XML Schema and WSDL contract, and finish by creating the necessary procedure code. The procedure code is created for you by TDV and is saved in the location specified.

Note: REST services do not have an associated WSDL contract.

Behavior of Abstract Contract-First WSDL

When you choose to use the abstract option with your contract-first WSDL, TDV automatically generates your input and output messages for you.

Behavior of Concrete Contract-First WSDL

When you choose to use the concrete option with your contract-first WSDL, TDV reads and uses the input and output binding messages that you have defined. The bindings that you have defined to control the security and transportation of the messages is retained and surfaced through Studio.

You can specify the port name for the binding.

To publish a contract-first WSDL and SOAP data service

1. Create a WSDL definition set. See Creating a WSDL Definition Set, page 384. For example, create NewWSDLDefinitionSet under the shared node of the TDV resource tree.
2. Make sure you are logged into Studio as a user with enough rights to publish resources.
3. Select the Web Services node under the Data Services node in the resource tree.
4. Right-click and select New Composite Web Service.

   The Add Composite Web Service dialog opens.
5. For Data Service Name, enter a name for your web service.
6. Click OK.
7. From the resource tree, open the web service that you just made.

   The Web service opens on the SOAP panel.

8. In the Service section, click Enable Contract First and set its value to true.

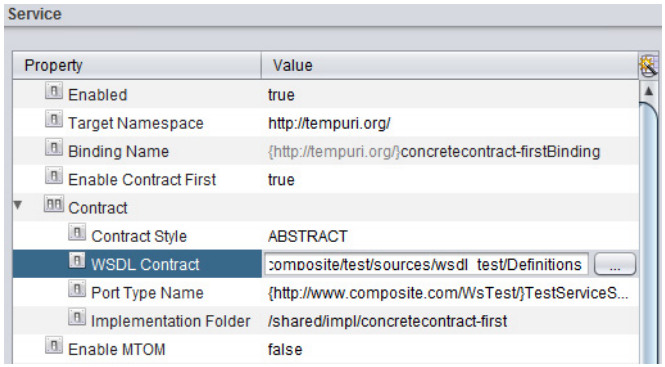
   Select true to point to your predefined WSDL and WSDL operations. After TDV retrieves your WSDL definitions, you must finish defining the WSDL operations using the Studio WSDL Definition Editor.

   A new property, Contract, appears below Enable Contract First.

9. Expand Contract.

10. Double-click WSDL Contract under Contract and type the resource path, or browse to your WSDL definition set.

    Visible if Enable Contract First is set to true. Type or use the browse button to specify the WSDL definition set that you have defined within TDV.



11. Save your Web service.

    Studio reads the WSDL definition set and populates the SOAP panel with the characteristics specified in the definition set.

12. In the Service portion of the screen, type or select values for the following properties:

    If the property has a default value, a value is required.

Properties	Description
Enabled	True (the default value) tells the system to use the information that you have defined on this tab.
Target Namespace	Displays the location of your SOAP installation. If your WSDL contract has a target namespace defined, it will be displayed in this field.
Binding Name	A unique binding name that can be referenced from elsewhere in the WSDL definition. This name signifies that the binding is bound to the SOAP protocol format: envelope, header and body.

Properties	Description
Contract	Property heading that is visible if Enable Contract First is set to true.
Contract Style	Visible if Enable Contract First is set to true. Select ABSTRACT or CONCRETE. Use concrete if your WSDL definitions contain bindings that you have defined to control the security and transportation of WSDL messages. If the WSDL that you have selected in the WSDL Contract field does not contain concrete bindings, you will not be able to select the CONCRETE Contract Style.
Service Name	Visible if Enable Contract First is set to true and if Contract Style is set to CONCRETE. Values are populated from the WSDL imported through your definition set.
Port Name	Visible if Enable Contract First is set to true and if Contract Style is set to CONCRETE. Select the port from the list of values. The port does not have to be unique within the WSDL, but it must be unique within the Web Service.
Port Type Name	Visible if Enable Contract First is set to true. Select the port from the list of values. The port does not have to be unique within the WSDL, but it must be unique within the Web Service.
Implementation Folder	Visible if Enable Contract First is set to true. Points to a TDV folder where the implementation procedures are generated. Double-click to edit the location.
Enable MTOM	Valid values are true or false. Set to true to enable the use of Message Transmission Optimization Mechanism (MTOM), a method of sending binary data to and from Web services.
Security Policy	Lists all available policies defined under localhost/policy/security.
SAML Validator Class	Type the name of the Java class that allows users to validate SAML assertions.
Subject Mapper Class	Define a Java class for your SAML content using TDV's extended Java API Subject Mapper, which can be found in: <TDV_install_dir>\apps\extension\docs\com\compositesw\extension\security\SubjectMapper.html Use method samlSubjectToCompositeUser to map samlSubject to an existing TDV user, so that this SAML principal has the same privileges as that user. After you create the Java class, save the JAR file to the <TDV_install_dir>\apps\server\lib folder.

Properties	Description
Endpoint URL Path	The path of the endpoint URL for this Web service. Default is the name (with spaces removed) you used for the Web service.
WSDL URLs	Property heading.
HTTP/SOAP 1.1	Displays the HTTP SOAP 1.1 URL that can be used to access the operation.
HTTP/SOAP 1.2	Displays the HTTP SOAP 1.2 URL that can be used to access the operation.
HTTPS/SOAP 1.1	Displays the HTTPS SOAP 1.1 URL that can be used to access the operation.
HTTPS/SOAP 1.2	Displays the HTTPS SOAP 1.2 URL that can be used to access the operation.

13. From the upper part of the Operations portion of the screen, select a Web service view or procedure that is available for use.

    This portion of the screen is automatically populated with the WSDL operations that were imported or defined in the WSDL definition set. If the definition set changes, you might need to save your Web service to see the latest list of WSDL operations. If you publish views or other procedures to the Contract First Web service, they should automatically appear in the Operations list.

14. To edit the Operations and Parameters portions of the screen, see the steps in Publishing a SOAP Data Service, page 422.

15. In the lower part of the Parameters portion of the screen, if you have chosen to define a concrete WSDL, you can view some additional properties and values of the chosen parameter.

    Element Name and Message Name for concrete WSDLs can be viewed, but not edited, from this Studio editor.

16. Save your definition.

Moving the Namespace Declaration From Elements to the Soap Header

In a published SOAP web service, to move the namespace declaration out of each element and define the namespace in the SOAP header you can use a Studio configuration parameter.

To toggle the location of the namespace declaration

1. From Studio, select Administration > Configuration from the Studio Modeler toolbar to open the Configuration window.
2. Locate the Declare Namespace in SOAP Envelope Element configuration parameter.
3. Set the value to true for the namespace to be declared once in the header or set it to false to have the namespace defined in each element.
4. Click OK to save changes and close the window.

For Example

When the value of the configuration parameter is false, each element would have something like this:

```
<soap-env:Envelope
xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Header>
<ns1:TestSQLScriptCustomevalueOutput
xmlns:ns1="http://tempuri.org/">custom</ns1:TestSQLScriptCustomeva
lueOutput>
</soap-env:Header>
```

And if the value of the configuration parameter is changed to true, then it is declared once as a namespace in the SOAP header

```
<soap-env:Envelope
xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://tempuri.org/">
<soap-env:Header>
<ns1:TestSQLScriptCustomevalueOutput>custom</ns1:TestSQLScriptCust
omevalueOutput>
</soap-env:Header>
```

Publishing a REST Service

This section describes how to add a REST-compliant data service. The configuration steps are optional.

- Configuration Options for Publishing a REST Service, page 432
- Publishing a REST Data Service, page 434

Configuration Options for Publishing a REST Service

This setting determines whether to enclose an XML output parameter with a wrapper element in a TDV REST Web Service when the parameter is a generic XML type.

To configure XML output parameter wrapper elements

1. From the Administration menu, choose Configuration.
2. Search for or navigate to the Generic XML REST Output Parameter Wrapper parameter.
3. Set the value to False to exclude the wrapper element, or set the value to True to include the wrapper element.
4. Stop and restart the TDV server to ensure that the parameter value change is recognized.
5. Make sure that the procedure that you define has a variable of data type 'XML'.

For Example

Assume that the Output parameter for the following code snippet is cust_data and that the data type is a generic XML type.

```
<ns1: cust_addr>
<mail_addr>
. . .
</mail_addr>
</ns1:cust_addr>
```

Set the Generic XML REST Output Parameter Wrapper parameter to true, to obtain output similar to:

```
<cust_data>
<ns1: cust_addr>
<mail_addr>
. . .
</mail_addr>
</ns1:cust_addr>
</cust_data>
```

If you do not want the parameter name wrapper in your output, set the configuration parameter to false. For example:

```
<ns1: cust_addr>
<mail_addr>
. . .
</mail_addr>
</ns1:cust_addr>
```

Publishing a REST Data Service

To publish a REST data service

1. Select the Web Services node under the Data Services node in the Studio resource tree.
2. Right-click and select New Composite Web Service.
3. For Data Service Name, enter a name for your Web service.
4. Click OK.
5. From the resource tree, open the data service that you just created.

   By default, the SOAP tab is displayed.
6. Optionally, publish views and procedures to the Web service:
   a. Select one or more tables, view, or procedure from the resource tree, and right-click.
   b. Select Publish.
   c. In the Publish window, type the name to use for the published resource.
   d. Select the Web service to which you want the resource to be published.
   e. Click OK.
7. Select the REST tab.
8. In the Service portion of the screen, type or select values for the following properties.

   If the property has a default value, a value is required.

Property	Required	Description
Enable	Yes	True (the default) tells the system you want to use the information defined on this tab.
Target Namespace	Yes	Replace the default value with the location of your REST installation.
Service Name	Yes	The name to give your service.

Property	Required	Description
Enable SSL/TLS	Yes	• Require SSL/TLS — True/false • Require Client Certificate — True/false SSL uses digital certificates, trusted certificate authority (CA), and encryption keys. When this is set to true, HTTPS/JSON and HTTPS/XML endpoints URLs are added to the Property list in the lower part of the Operations section.
JSON Context Url Pat		Default value is /json.
XML Context Url Path		Default value is /xml.

JSON Format	Yes for JSON	• JSON Package Name — Default name of your JSON service. The input and output data use this package name.
		• Use Parameter Name of Procedure— If true, a JSON object name with the format of parameter will be the same as the original name defined in the procedure. The default value is false. If false, the format is:
		`<package name> + '.' + <procedure name>+<parameter name>`
		• Wrapped Cursor—If true, a JSON row object is added as a wrapper of the cursor output, for tables and procedures.
		• Wrapped Table—If true, the table output is wrapped with an output JSON object with a pattern of:
		`<package name> + '.' + <table name> + 'output'`
		For the wrapped options, the default value is true. Default value is consistent with an XML definition.

Enable HTTP Basic	Yes	Valid values are true or false. Set to true to enable a method to provide a username and password (when making a request) that results in a string encoded using a Base64 algorithm.
Enable HTTP NTLM	No	Used to enable NTLM authentication over HTTP.
Enable HTTP Negotiate	No	Used to enable Kerberos and Active Directory authentication over HTTP.

9. From the upper part of the Operations section, select a Web service view or procedure that is available for use.

   If you publish a new view or procedure to the Web service while this window is open, you might need to close and reopen this window for it to appear in the list.

10. In the lower part of the Operations section, type or select values for the properties listed in the following table.

Property		Description
HTTP Method		Select the verb associated with the procedure, for example GET, POST, PUT, or DELETE. Depending on the choices you have made for the operations field, these options might already be defined.
Operation URL Path		Specify the operation URL path as defined for your operation. Follow endpoint URL guidelines to configure your values. You can use brace tags. You can specify a portion of the endpoint URL to be used in each Web service module. In a published WSDL file, the URL defining the target endpoint address is found in the location attribute of the port's soap:address element.
Input Message		
	Parameter Style	Use this field to determine the shape of the input argument or message. This value applies to all parameters. • Select WRAPPED when there might be multiple parameters that use the BODY location. The wrapper element is the child of the REST BODY and the parameter elements are children of the wrapper element. • Select BARE when exactly one parameter has a location of BODY and its element is the only child of the REST BODY. All other parameters of the same direction must be mapped to another location—for example, HEADER.
	Wrapper	
	Element Name	Unique name for the wrapper element. For example, if you type wrappedOrderParams here, the WSDL XML includes an element: <wrappedOrderParams> </wrappedOrderParams>
	Element Type	This is the type value of your row element.
Output Message		

Property		Description
	Parameter Style	The value applies to all parameters. • Select WRAPPED when there might be multiple parameters that use the BODY location. The wrapper element is the child of the REST BODY and the parameter elements are children of the wrapper element. • Select BARE when there is exactly one parameter with a location of BODY and its element is the only child of the REST BODY. All other parameters of the same direction must be mapped to another location, for example, HEADER.
	Wrapper	
	Element Name	Unique name to give to the wrapper element. For example, if you type wrappedOrderParams here, the WSDL XML includes an element: <wrappedOrderParams> </wrappedOrderParams>
	Element Type	This is the type value of your row element.
Endpoint URLs		
	HTTP/JSON	Displays the HTTP/JSON endpoint URL for this operation.
	HTTPS/JSON	Displays the HTTPS/JSON endpoint URL for this operation. Present only if Enable SSL is true in the Service section of this panel.
	HTTP/XML	Displays the HTTP/XML endpoint URL for this operation.
	HTTPS/XML	Displays the HTTPS/XML endpoint URL for this operation. Present only if Enable SSL is true in the Service section of this panel.

11. Save your settings.
12. Repeat the steps in Publishing a SOAP Data Service, page 422 for each operation or view you want to define for the Web service.
13. If your operation has parameters, go to the Parameters portion of the screen and select a parameter from the upper list.

14. In the lower part, type or select values for the following properties for input or output parameters.

Property	Resource Type	Description
Name	All	Name of the parameter. Cannot be edited.
Row Element Name	Table View	Fully qualified name to give to the input parameter, output parameter, or output cursor; or name (relative to the cursor path) to give to the column.
Row Type Name	Table View	(Cursor only) Type value of this row element.
Cursor Type Name	Table View	(Cursor only) Unique name for the type that defines the global cursor element.
Element Name	All	Can be NULL. The default value of the Element Name field specifies the schema of the REST message that is encrypted. This procedure can be used to encrypt the REST message body.
Binding Location	Procedure	When the selected operation is a procedure, you can define binding locations. If the message is BARE, exactly one parameter must use the BODY binding location. The available locations for input are: ENTITY (not available for GET), QUERY, HEADER The available locations for output are: ENTITY
Query	Procedure	The parameter-name part of a query string. The query argument can accept NULL values. For example, the following query arguments are both valid: • http://localhost:9410/json/goody/data?name_arg={name_arg_value} • http://localhost:9410/json/goody/data
Default Value	Procedure	Optional. Default value to use for the parameter if a parameter value is not passed through the request.
Nullable	Procedure	Whether this field is allowed to have no value. True specifies that the parameter can be passed to the server with no value. Without a value for the parameter, all data is returned from the query.

Property	Resource Type	Description
Element Name	Procedure	Can be NULL. The default value of the Element Name field specifies the schema of the REST message that is encrypted. This procedure can be used to encrypt the REST message body.
Direction	All	Indicates the direction of the input parameter, output parameter, or output cursor. Cannot be edited.

15. Repeat the steps for each parameter to be defined for the Web service.
16. Save your definition.

Web Services Security

TDV offers Web service security at two levels: transport-layer level and message level. Secure Web services are published by TDV using Web Services Security (WSS). TDV enables signing of messages with digital certificates to identify the source of the message, and encryption of the message for secure delivery to the client.

- Supported Web Service Security Standards, page 439
- Using a Predefined Security Policy for a Web Service, page 441
- Creating and Using a Custom Security Policy for a Web Service, page 442

Supported Web Service Security Standards

The following security policies, in the form of XML files, are provided for Web service clients.

Note: The transport level security policies (http basic and https basic) can be applied to a SOAP Web service, but they do not prevent an anonymous user from invoking the Web service. When using the basic security policies or when creating a custom security policy, consider this security issue.

Transport or Standard	System Security Policy	Description
HTTP	Http-Basic-Authentication.xml	Policy that requires a username and password when making a request.

Transport or Standard	System Security Policy	Description
HTTP	Http-Negotiate-Authentication.xml	Policy that enables Kerberos authentication.
HTTP	Http-NTLM-Authentication.xml	Policy that enables NTLM authentication.
HTTP	Http-UsernameToken-Digest.xml	Policy that validates against a UsernameToken header encrypted using a nonce value.
HTTP	Http-UsernameToken-Plain.xml	Policy that validates against a UsernameToken header. The password can be in plain text.
HTTPS	Https-Basic-Authentication.xml	Policy that requires a username and password when making a request.
HTTPS	Https-ClientCertificateRequire.xml	Policy that requires client certificates.
HTTPS	Https-UsernameToken-Digest.xml	Policy that validates against a UsernameToken header encrypted using a nonce value.
HTTPS	Https-UsernameToken-Plain.xml	Policy that validates against a UsernameToken header. The password can be in plain text.
SAML	Saml1.1-Bearer-Wss1.1.xml	Method in which the bearer assertion is used to facilitate single sign-on to the web browser.
SAML	Saml1.1-HolderOfKey-Wss1.0.xml	Method that establishes a correspondence between a SOAP message and the SAML assertions added to the SOAP message.
SAML	Saml1.1-SenderVouches-Wss1.1.xml	Subject-confirmation method that enables an attesting entity to vouch for the identity of a subject to a party that trusts the sender.

The following legacy WSS standards are supported:

- OASIS Web Services Security: SOAP Message Security 1.0 Standard 200401, March 2004
- Username Token profile V1.0
- X.509 Token Profile V1.0

The TDV Server accepts certificate formats such as X.509 and JKS so that keys and message signatures do not need to be converted to ASCII before sending.

The TDV pipeline implementation allows SAML and Kerberos tokens to be passed through to message destinations, but TDV Server does not directly process or use SAML and Kerberos tokens.

WSS incorporates security features in the header of a SOAP message, and works in the application layer to help ensure end-to-end security for SOAP messages.

WSS UsernameToken SOAP header validation is independent of the message-security pipeline; the message security pipeline cannot process these contents.

Transport Layer Security (TLS) is supported to ensure message integrity and confidentiality through HTTPS, reducing performance overhead. If the messaging needs to pass through a proxy server, however, TLS should not be used without special handling considerations.

Using a Predefined Security Policy for a Web Service

Security policies for Web services require a definition using the security protocol of your choice. TDV provides a generic editor that you can use to type your security policy, or you can import security policy files from anywhere in your network. Predefined and custom security policies populate the list of security policies available for defining a SOAP Web service.

After defining your policy, you can use the policy in your new or existing SOAP Web services. For more information, see Publishing a SOAP Data Service, page 422.

Note: The transport level security policies (http basic and https basic) can be applied to a SOAP Web service, but they do not prevent an anonymous user from invoking the Web service. When using the basic security policies or when creating a custom security policy, consider this security issue.

To use a TDV security policy

1. Open Studio.
2. Expand localhost/policy/security/system.
3. Review the list of predefined policies under system and determine the policy to use.

   For descriptions of the available policies, see Supported Web Service Security Standards, page 439.
4. Open a published SOAP Web service.

   For more information, see Publishing a SOAP Data Service, page 422.
5. Select the SOAP tab.

6. Left-click in the Value column next to Security Policy in the Services or the Operations part of the SOAP tab and choose one of the predefined or custom security policies from the drop-down list.
7. Save your changes.

Creating and Using a Custom Security Policy for a Web Service

Security policies for Web services require a security protocol definition. You can import security policy files, or define a custom policy using the generic editor that TDV provides.

To create a custom security policy within TDV

1. Open Studio.
2. Expand localhost/policy/security/user.

   To review the security policies that TDV provides, expand localhost/policy/security/system.
3. Right-click on user and select New Policy.
4. Type a name for the security policy.
5. The security policy editor opens in the workspace pane on the Policy tab.
6. Type or import your security policy.

To import your security policy, click Insert from File and browse to the file.

7. Edit the policy as needed.
8. Optionally, click Format the Policy to formatted it according to XML standards.
9. Optionally, add annotations to the policy on the Info tab.
10. Open a published SOAP Web service.
11. Save the new policy.

    If your policy definition is invalid, your security policy is created, but the invalid code is not saved. To validate what code was saved for a security policy, close the editor and reopen your policy from the resource tree.

    For more information, see Publishing a SOAP Data Service, page 422.
12. Navigate to the SOAP tab.
13. Left-click in the Value column next to Security Policy in the Services or the Operations part of the SOAP tab and choose your custom security policies from the drop-down list.

14. Save your changes.

Security and Privileges on Published Resources

Using a published resource requires appropriate access privileges on both a published resource and the original resource. A published resource has its own READ and WRITE privileges to let you manipulate it in Studio, but it does not have its own run-time privileges (EXECUTE, SELECT, INSERT, UPDATE, DELETE). You can set up privileges for the published and shared areas in Studio to suit the needs and restrictions of your environment. For more information on how to set up various privileges, refer to the *TDV Administration Guide*.

Privileges on objects in the Data Source are evaluated only at the time of introspection or later when a FETCH is called during statement execution. So if privileges on an object in the data source are lowered after introspection, the error will not be caught until actual statement execution.

Column-based security privileges are set up the same way as for any other object exposed through TDV. If you restrict access to a view in the published layer, the shared area, and in the introspected data source, the column would have the same restrictions. Similarly if you mask the data in a column for a view, that data is masked in the same way for the published data.

How resource security and privileges work for a given TDV environment can vary. It is best to test several scenarios to make sure that you can access resources as expected. You can publish a table as a database, as a view through a web service, or as a view through a legacy web service, so you should check all scenarios.

When you run a SELECT * from an external client, the likely results are:

- An error message states that you do not have appropriate privileges on a column in your query.
- You get a result that contains ONLY the columns that you have privilege to see.
- The columns for which you do not have privileges have all NULL values.

You can control which of these outcomes occurs using a configuration parameter.

Controlling the Outcome of Trying to Access a Column without Permissions

You can control which of the outcomes occurs using a configuration parameter.

To control what is returned for a column for which the user lacks

appropriate permissions

1. From the Studio main menu, select Administration > Configuration.
2. In the Configuration window, navigate to Server > Configuration > Security > Enable Exception For Column Permission Deny.
3. Set this parameter to true if you want to raise an exception when column-level permissions are missing.
4. Set or leave this parameter set to false, if you want columns for which the user does not have permissions to contain NULLs or not be returned at all.

Disable OData for Published Data Sources

You can globally disable OData for all published data sources.

To disable OData for published data sources

1. From the Studio main menu, select Administration > Configuration.
2. In the Configuration window, locate the Enable OData Services parameter.
3. Set this parameter to false to disable OData for published data sources.
4. Click Apply and OK to save the setting and exit the configuration dialog.

Exploring Data Lineage

This topic describes how to display and explore the data lineage from the data sources through to the published resources defined in TDV.

About Data Lineage

When you work with resources in Studio, there are often many relationships and dependencies between resources. A composite view, for example, depends on things like data sources and procedures which might then depend on a different data source. Other views might reference the view and thus be dependent on it. Security policies, triggers, and other resources might be dependencies. Understanding these interdependencies can be useful in analyzing where data comes from, how it is derived, and how it is used.

To help you understand resource dependencies, Studio provides a data lineage graph that supports TDV modeling and data visualization capabilities. The TDV data lineage feature helps you:

- Trace where data originates—From data sources through published resources, you can trace which specific data sources contribute to each column.
- Understand how data is used and/or transformed—You can determine how the data is used or transformed by TDV resources and which data contributes to the published resources.
- Discover the impact of change and analyze dependencies—If you delete or rename a resource, the data lineage graph for related resources graphically indicates which dependent resources are impacted.

You can obtain the data lineage for all resource types, but lineage for these resources can be especially useful:

- Data sources
- Tables
- Composite views
- Models
- Procedures
- Definition sets
- Published services
- System resources

See Lineage Information for Different Resource Types, page 461 for what is displayed for each resource type along with examples.

Determine Where Data Comes From

The data lineage graph helps you discover:

- What data sources contribute to each column? The data lineage graph helps you understand how many systems a view is hitting and if more data is requested, where TDV gets it.
- What are the platforms for each data source? This can help with optimizations like push down, data ship, and caching.
- How many records are pulled from each data source? Combine some of the query plan functionality with data source lineage to help with data ship optimizations.

Determine How Data Is Used

For any defined data source, you can open a data lineage graph that shows all TDV resources that use that data. You can then expand the resources in the lineage graph to see the columns that are participating in the lineage.

Determine How Data Is Transformed

When data is modified by procedures or transformed, the data lineage graph can help you trace the resources involved and how the changes are propagated. From the data lineage graph, you can easily open the editor for any of the resources to discover exactly what the changes are. For example, you can:

- See that a projection is concatenated in a prior view, then CAST to a date, then used in a GROUP BY clause.
- See that a column is used in a JOIN or is used in a filter.
- See that a column is used in aggregate or analytical function.

Analyze Changes and Dependencies

The data lineage graph helps you determine:

- What will be affected by a proposed change (rename, deletion, addition) to a column or resource? Upstream references are important in this use case. This includes what calls/references the column might need to also be changed (if for example a rename of the column is performed). Also important is where the column is published since that impacts external systems that could break with a change.
- How published resource/columns are provided to a consuming application and debug any issues by analyzing the downstream dependencies back to the originating system.
- What resources you need to have privileges to because of the resource dependencies. See "About Managing Dependency Privileges" in the *TDV Administration Guide* which says that a user/group must have the privileges to access all dependencies. The lineage graph can help you investigate the resources to which you need privileges.

Lineage Panel Buttons and Controls

The following table lists the unique controls in the **Lineage** panel and what they do.

Label	Use to...
Show Detail	Display the definition of the resource in a new panel beneath the resource lineage graph. This button does not appear on the **Lineage** panel when it is opened from within a resource editor.
Save to File	Open a dialog box to save the dependencies in a file. See Exporting Lineage Reports to a Comma-Separated-Values File, page 459.
Hide/Show Dependencies	A toggle button that hides or shows dependencies for a selected resource when enabled. By default, dependencies are displayed.
Hide/Show References	A toggle button that hides or shows references for a selected resource when enabled. By default, references are displayed.

Label	Use to...
Hide/Show Cached Tables/Views	A toggle button that hides or shows cached tables and views for direct lineage. By default, cached tables and views are displayed.
Show/Hide Indirect Links	A toggle button that shows or hides WHERE/GROUP BY/FROM and other indirect links when enabled. By default, indirect links are hidden.

Displaying Data Lineage

You can display the data lineage for most TDV resource types. The kinds of interdependencies displayed depend on the resource type as illustrated in Lineage Information for Different Resource Types, page 461.

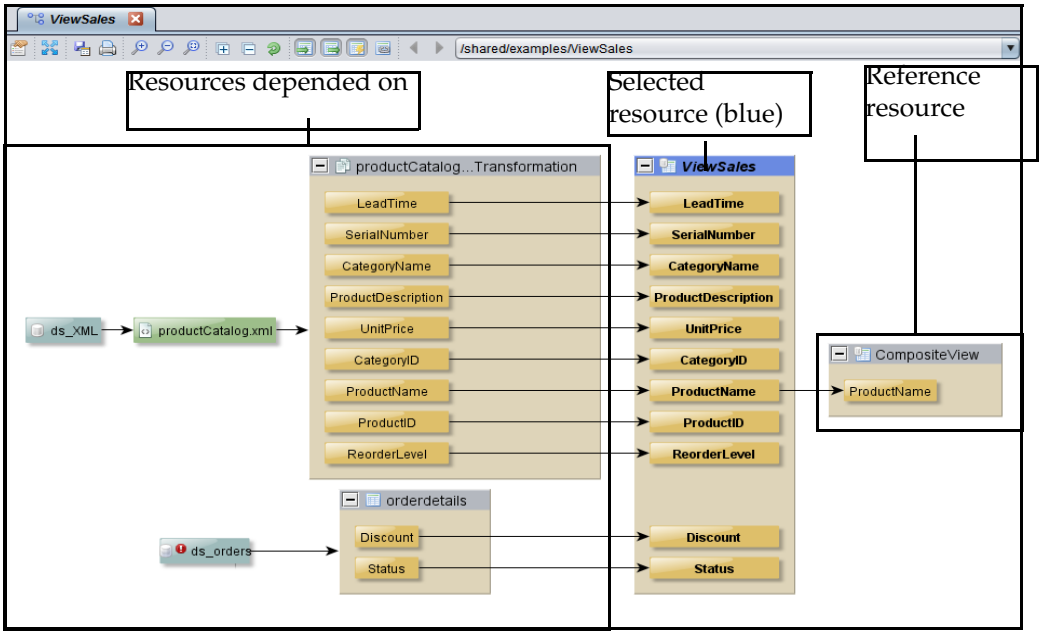
To display data lineage

1. Select a resource in the resource tree.
2. Open the data lineage panel by:
   - Choose **Open Lineage** from the **File** menu.

   Or

   - Right-click and choose **Open Lineage** from the popup menu.

Studio displays the lineage for the selected resource including all resource dependencies and all reference resources. This example shows the data lineage for the ViewSales example view:



The data lineage graph shows:

- Where the data originates.
- What types of resources are dependences and references.
- What resources the current resource depends on.
- What resources reference the current resource.
- How a resource is derived from other resources.
- What resources are impacted, if any. Impacted resources are indicated with an icon. See Impacted Resources, page 47.
- Circular references, if any.
- See which resources have security policies. The security content is not displayed but can be viewed if you have the necessary credentials.

See Working with the Data Lineage Graph, page 450 for how to use the lineage graph.

Working with the Data Lineage Graph

The data lineage panel gives you a relatively large area in which to graphically view all of the dependencies and dependent resources. See Lineage Panel Buttons and Controls, page 447 for the purpose of the various buttons and controls in the data lineage panel.

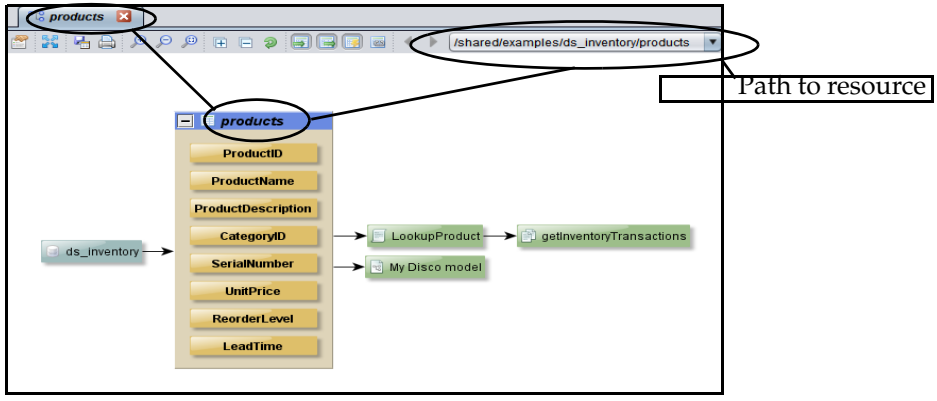
You can navigate and explore the relationships and resource dependencies as follows:

To ...	Do this...
Change the focus to a different resource and display its lineage.	Double-click a resource or use the right-click menu. See Changing the Resource Focus, page 451 for more information.
Expand and collapse what is displayed.	Use the expand and collapse buttons to see a resource's contents. See Changing What Is Displayed, page 452.
Filter the visible columns	Use the funnel button on the view title bar. See Filtering the Visible Columns, page 453.
Go back to a previous data lineage display.	Use the navigation arrow buttons or select a resource in the history drop list. See Using the History Feature, page 456 for more information.
Open a resource's editor in another panel.	Right-click the resource and choose **Open**.
Get quick information about a resource.	Hover over the resource to display a popup. See Getting Resource Summary Information, page 453 for more information.
Get details about the resource definition.	Click the **Details** button. Displays the resource definition in the Detail beneath the lineage graph. See Getting Resource Details, page 454 for more information.
Print the lineage in a regular size or poster print.	Click the **Print** button. See Printing the Lineage Graph, page 457 for more information.
Discover impacted resources.	Impacted resources are indicated with a warning icon. You can hover over the resource to get more information. See Impacted Resources, page 47 for more information.

To ...	Do this...
Refresh the graph.	See Refreshing the Lineage Graph, page 457 for more information.
Export lineage information to a .csv file	See Exporting Lineage Reports to a Comma-Separated-Values File, page 459.

Changing the Resource Focus

When you view a lineage graph, the resource on which the lineage graph is based is displayed with a blue title bar. From this graph, you can change the focus to another resource to get more information about the dependencies and relationships.



To change the resource focus

1. Open the lineage for a resource.
2. Change the focus by either of these:
   - Double-click any resource in the graph to view its lineage.

     Studio displays the lineage for the selected resource. Its title bar is blue and the lineage panel tab shows the name of the new resource.

     Note: The history list is updated and you can return to any previous lineage display.
   - Right-click any resource and choose **Open** to open the resource's editor in a new tab.

Changing What Is Displayed

The toolbar buttons let you change what information is displayed on the lineage graph as described in these sections:

Expanding and Collapsing the Lineage Graph

- Use the **Expand All** button display column details.
- Use the **Collapse All** button hide column details.

  The default is expanded.

Hiding and Showing Dependencies

- Use the **Hide/Show Dependencies** button to hide or show dependencies.

  The default is to show dependencies.

Hiding and Showing References

- Use the **Hide/Show References** button to hide or show references.

  The default is to show references.

Hiding and Showing Cached Tables and Views

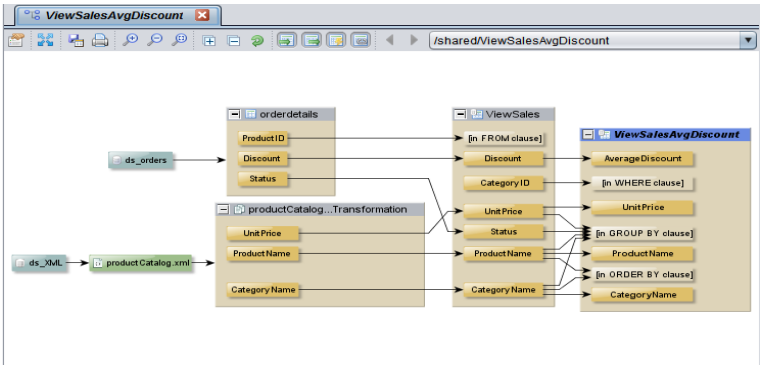
- Use the **Hide/Show Cached Tables/Views** button to hide or show cached tables and views.

  The default is to show cached tables and views within the direct path of lineage.

Hiding and Showing Indirect Links

- Use the **Hide/Show Where/Group By/From and other Indirect Links** button to hide or show indirect links which are references in a view to a column by a WHERE, GROUP BY, ORDER BY, FROM, HAVING, or other non-SELECT clause.

The default is to hide indirect links. The following example shows four indirect links that reference specific columns.

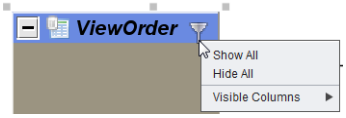


Filtering the Visible Columns

You can filer the columns that are visible to a select group or hide/show them all.

To filter the visible columns

1. In the Lineage panel, click the funnel button on the title bar of the selected view.



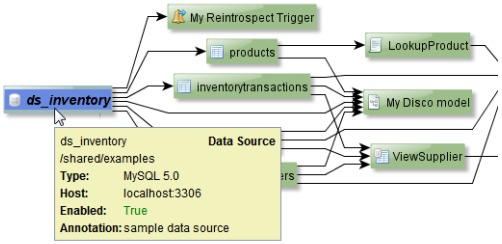
2. Select one of the filter options:
   - Show All—displays all of the columns in the view.
   - Hide All—Displays no columns in the view.
   - Visible Columns—Select or uncheck the columns to be displayed. Note that the visible columns are listed in alphabetical order which might not match the order in the display.

Getting Resource Summary Information

You can obtain summary information about any resource in the lineage graph. You can tell the resource type by the icon next to its name.

To display resource summary information

1. Hover your mouse over the resource.
2. Studio displays a tooltip with information about the resource.



This is the same information displayed in the resource tree tip when you hover over a resource.

You can also right-click any resource and choose **Open** to open the resource's editor in a new tab.

Getting Resource Details

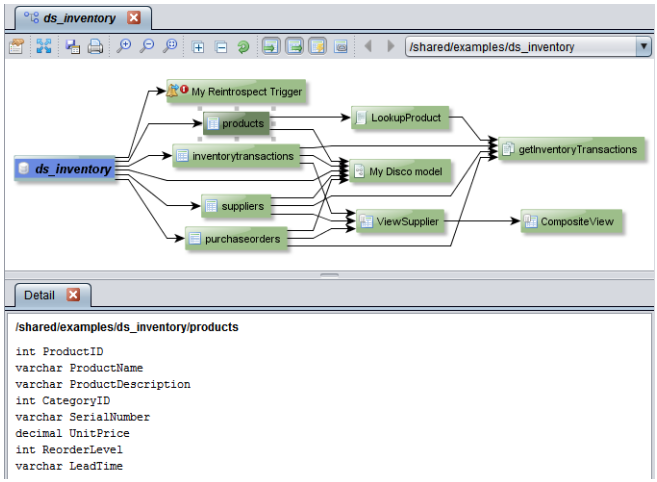
You can get detailed information about any resource in the lineage graph. For example, for tables you can display the data types for each column. For views, you can view the SQL that defines the view.

You can tell the resource type by the icon next to its name.

To display resource details

1. Open a lineage graph.
2. Optionally, select any resource in the lineage graph.

3. Click the **Show Detail** button.



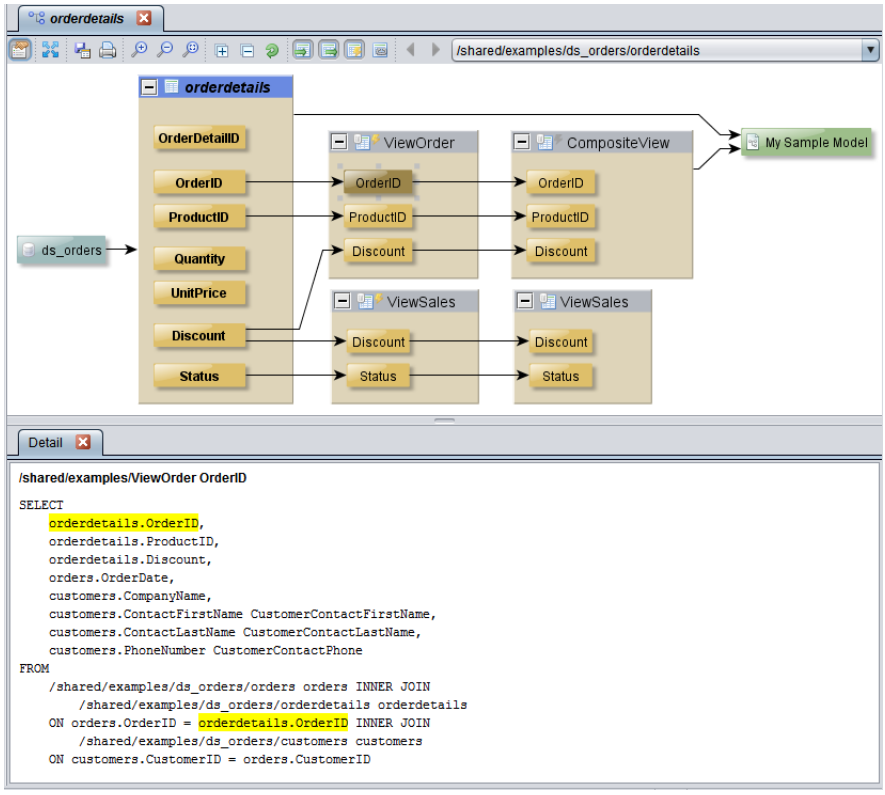
4. Optionally, select a different resource to display its details in the same tab.

   The type of information displayed depends on the resource type:

Resource Type	Details Provided
Data source	Resource path in TDV.
Script	Code that defines the script.
Table	Column data types.
Transformation	Code that defines the transformation.
View	SQL that defines the view. A selected column is highlighted in yellow.

For example, when you double-click a view to display its columns, the columns and their relationships are displayed and the **Detail** panel shows the

SQL that defines the view. If you select a column, then the references to the column are highlighted in yellow as shown in this example:

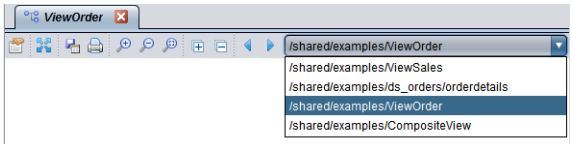


Using the History Feature

When you change the resource focus, Studio adds the previous focus to the history list so that you can easily return to a previous view. This history list is saved with the resource and can be used again if you reopen the resource's lineage graph.

To display a previously viewed lineage graph

1. Open the lineage graph for a resource.
2. At the top right corner of the resource graph, click the drop-down arrow next to the path for the current resource.



3. Select the path for the resource you want to be the central focus.

   Studio displays the resource graph that was previously viewed.

Note: You can also use the **Navigate Backward** and **Navigate Forward** arrow buttons to step through the lineage graphs.

Refreshing the Lineage Graph

The lineage graph displays what is currently saved to the TDV Server. If you make changes and save them, you can update the lineage graph. The print **Preview** dialog lets you control the size of the print, crop it, print the graph as a poster, add a title, and add a footer.

Note: If you have unsaved changes when you open a lineage graph, Studio displays a warning message to alert you. If you do not want to see these messages, you can turn this feature off using the Studio **Options** dialog; uncheck the **Warn before data lineage calculation when resource editor is saved** option on the **Warnings** tab.

To refresh the lineage graph

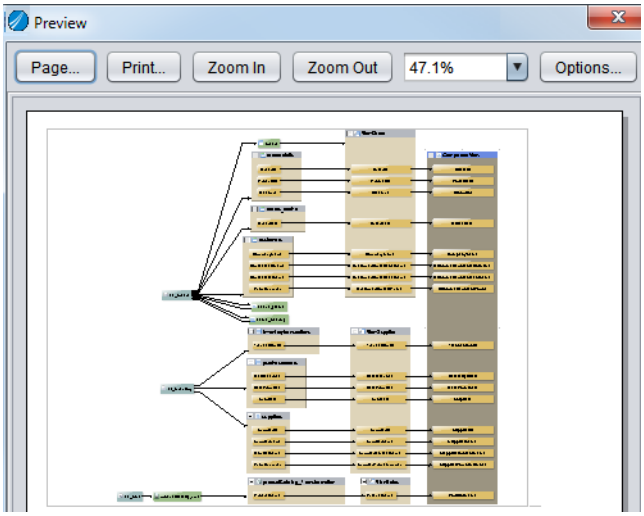
1. Display the resource lineage graph.
2. If necessary, save any changed resources involved in the lineage graph.
3. Click the **Refresh** button.

Printing the Lineage Graph

Studio lets you print the lineage graph and provides various ways to control the printed output like creating a large poster print, adjusting the orientation and margins, and adding a title and a footer.

To print a lineage graph

1. Display the resource lineage graph.
2. Click the **Print** button.



Note: If your lineage graph is especially large, you can create a poster print of it using the **Options** button.

3. Use the buttons at the top control the printed output as follows:

Use...	To...
Page	Change the orientation between portrait and landscape and to adjust the margins.
Print	Print the model diagram using your standard print dialog.
Zoom In	Zoom in on the print image. This does not increase the size of the image on the page.
Zoom Out	Zoom out on the print image. This does not decrease the size of the image on the page.
Zoom amount	Select a setting from the drop-down list.

Use...	To...
Options	Displays a Print Options dialog with three tabs: • General tab options let you create a large poster-style print of the model diagram by dividing the model into sections that can be pieced together. Poster Rows—divides the image vertically into the number of rows entered and displays how it would appear on that number of pages if printed. Poster Columns—divides the image horizontally into the number of columns entered and displays how it would appear on that number of pages if printed. Add Poster Coordinates—prints the location of each subsection of the model diagram so that you can figure out how the sections go together. Clip Area—Two options are provided: Graph enlarges the model image as much as possible within the margins; View increases the blank space around the model image. • Title tab options let you specify a title for the printed model. Text—Specify the title you want to appear at the top of the model diagram. Titlebar Color—Click to select the color of the title bar which will appear behind the title text. Text Color—Click to select the color of the title text. • Footer tab options let you specify a footer for the printed model. If you are printing the model poster style, the footer will span the pages appropriately. You can specify text, footer and text color.

Exporting Lineage Reports to a Comma-Separated-Values File

All of the information represented in the tree view and the graph view of the dependency report can be exported to a comma-separated-value (csv) file.

To export dependency reports to a csv file

1. Display the resource lineage graph.
2. Make sure that the lineage graph has the resource focus that you want.

   The lineage report in the csv file is based on the current resource which is shown with a blue title bar.

   Note: It does not matter if the lineage graph is expanded or collapsed. Column information is included in the exported file, regardless. If you hide

dependencies or references in the lineage graph, they are not included in the exported file.

3. On the lineage button toolbar, click the **Save To File** button.
4. In the **Save** dialog, navigate to the location where you want to save the information.
5. Name the file that will contain the dependency information.
6. Click **Save**.
7. Open the file in Excel or other similar application.

   The lineage report contains one row for every connecting line in the lineage graph. A connecting line represents a dependency or reference. The information for each dependency or reference is as follows:

SourcePath	SourceType	SourceColumnName	ReferenceByPath	ReferenceByType	ReferencedByColumnName
The path to source	The source type: TABLE, DATA_SOURCE, PROCEDURE, MODEL, and so on	For column dependencies, the name of the column.	The path to the reference.	The resource type: TABLE, DATA_SOURCE, PROCEDURE, MODEL, and so on	For column references, the name of the column.

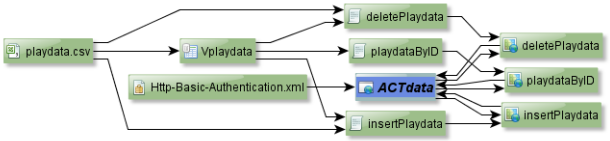
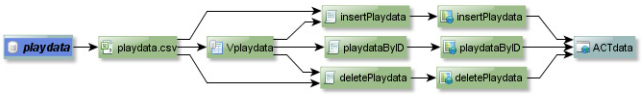
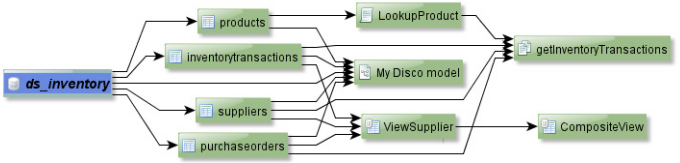
Sample lineage information:

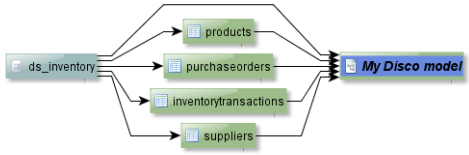
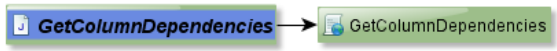
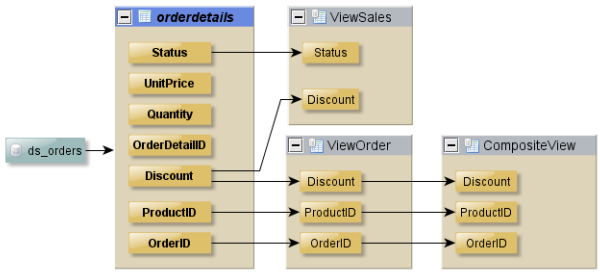
A	B	C	D	E	F
ı	SourceType	SourceColumnName	ReferencedByPath	ReferencedByType	ReferencedByColumnName
amples/CompositeView	TABLE		/shared/My Sample Model	MODEL	
amples/ViewSupplier	TABLE		/shared/examples/CompositeView	TABLE	
amples/ds_inventory/suppliers	TABLE		/shared/examples/ViewSupplier	TABLE	
amples/ds_inventory	DATA_SOURCE		/shared/examples/ds_inventory/suppliers	TABLE	
amples/ds_inventory/inventorytransactions	TABLE		/shared/examples/ViewSupplier	TABLE	
amples/ds_inventory	DATA_SOURCE		/shared/examples/ds_inventory/inventorytransactions	TABLE	
amples/ds_inventory/purchaseorders	TABLE		/shared/examples/ViewSupplier	TABLE	
amples/ds_inventory	DATA_SOURCE		/shared/examples/ds_inventory/purchaseorders	TABLE	
amples/ViewSales	TABLE		/shared/examples/CompositeView	TABLE	
amples/productCatalog_Transformation	PROCEDURE		/shared/examples/ViewSales	TABLE	
amples/ds_XML/productCatalog.xml	TREE		/shared/examples/productCatalog_Transformation	PROCEDURE	
amples/ds_XML	DATA_SOURCE		/shared/examples/ds_XML/productCatalog.xml	TREE	
amples/ds_orders/orderdetails	TABLE		/shared/examples/ViewSales	TABLE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/orderdetails	TABLE	
amples/ds_orders/cache_status	TABLE		/shared/examples/ds_orders	DATA_SOURCE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/cache_status	TABLE	
amples/ds_orders/cache_tracking	TABLE		/shared/examples/ds_orders	DATA_SOURCE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/cache_tracking	TABLE	
amples/ViewOrder	TABLE		/shared/examples/CompositeView	TABLE	
amples/ds_orders/orders	TABLE		/shared/examples/ViewOrder	TABLE	
amples/ds_orders/orders_cache	TABLE		/shared/examples/ds_orders/orders	TABLE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/orders_cache	TABLE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/orders	TABLE	
amples/ds_orders/orderdetails	TABLE		/shared/examples/ViewOrder	TABLE	
amples/ds_orders/customers	TABLE		/shared/examples/ViewOrder	TABLE	
amples/ds_orders	DATA_SOURCE		/shared/examples/ds_orders/customers	TABLE	
amples/ViewSupplier	TABLE	SupplierName	/shared/examples/CompositeView	TABLE	SupplierName
amples/ds_inventory/suppliers	TABLE	ContactName	/shared/examples/ViewSupplier	TABLE	SupplierContactName
amples/ds_inventory/suppliers	TABLE	SupplierName	/shared/examples/ViewSupplier	TABLE	SupplierName
amples/ds_inventory/suppliers	TABLE	SupplierID	/shared/examples/ViewSupplier	TABLE	SupplierID
amples/ds_inventory/inventorytransactions	TABLE	TransactionID	/shared/examples/ViewSupplier	TABLE	TransactionID

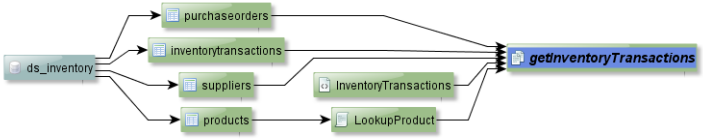
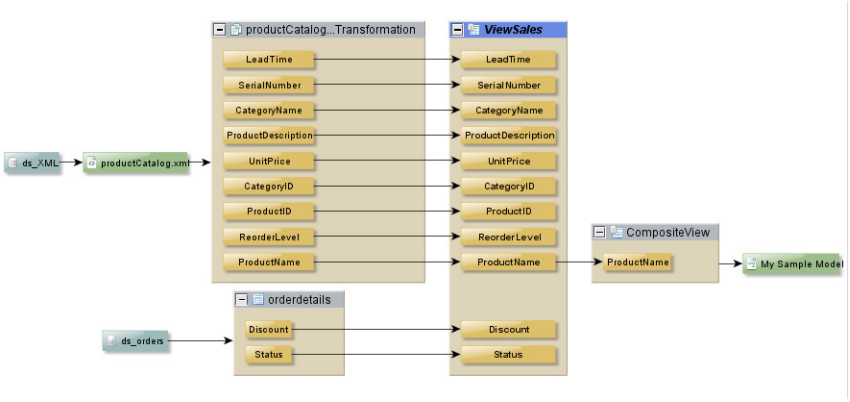
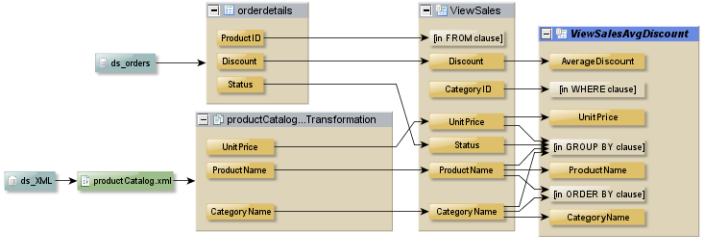
Lineage Information for Different Resource Types

You can obtain the lineage information for most resource types. The information is displayed graphically in the resource lineage graph. The relationships and connection between different resources can be especially useful. Also, you can change the resource focus to navigate between resources and understand their relationships.

This table illustrates examples of the lineage information that is displayed for many resource types (listed in alphabetical order).

Resource Type	Description and Example
TDV Web service	Displays all resources on which the Web service depends and all dependent published resources. For example: 
Data source (.csv)	Displays all resources that depend on the selected .csv data source. For example: 
Data source (relational)	Display all resources that depend on the selected relational data source. Example: 
Data source (XML)	Displays all resources that depend on the selected XML data source. For example: 
Definition Set	Displays all resources that depend on the selected definition set. For example: 

Resource Type	Description and Example
Hierarchical resource	Displays the data source on which the resource depends and all resources that depend on the selected hierarchical resource. For example: 
Model	Displays all resources on which the selected model depends. For example: 
Policy	Displays all resources that depend on the selected policy. For example: 
Procedure (Script)	Displays all resources on which the procedure depends and all resources that reference the script. For example: 
Published resource	Displays all dependencies for the selected resource. For example: 
Table	Display the parent data source and all resources dependent on a table and its individual columns. Example: 

Resource Type	Description and Example
Transformation (XQuery)	Displays the resources on which the XQuery transformation depends. For example: 
Transformation (XSLT)	Displays the dependencies and references for the XSLT transformation. For example: 
View	Displays the dependencies, references, and cached tables/views for a view (by default). For example:  You can also toggle the indirect links button to view the references to columns in a view's WHERE, GROUP BY, HAVING, FROM, or other non-SELECT clauses as shown here: 

TDV Caching

Caching makes a copy of frequently accessed data and stores it for quicker, more convenient access. The following topics are covered:

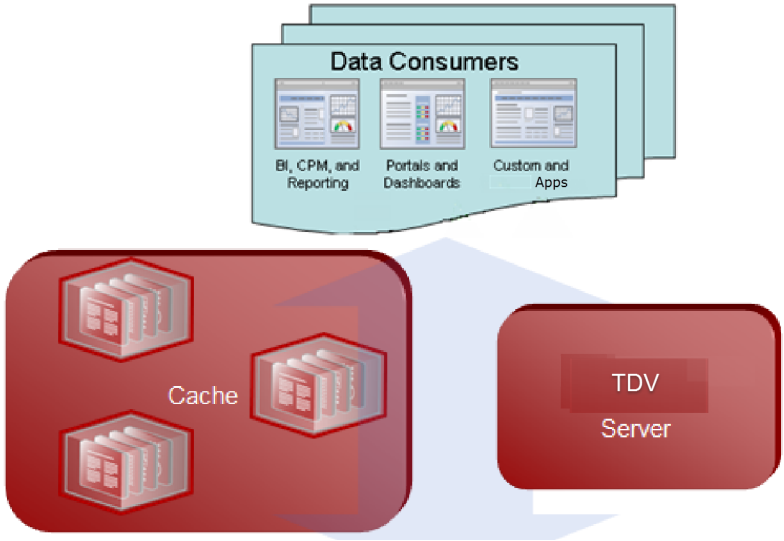
Overview of TDV Caching

TDV caching is a feature that can improve the responsiveness of data access in your enterprise. Caching features can help you quickly build data caches that complement data virtualization and improve reporting. Working within TDV and Studio, you can access data and metadata from across data repositories, whether they be applications, databases, or data warehouses. You can then use TDV to create data caches. From the data caches, you can consume the cached data for client activities such as data analysis.

TDV caching is a feature of TDV and uses the Studio user interface. TDV caching is delivered with TDV, and contains all of the software needed to implement TDV caching in your IT environment.

- TDV Caching Concepts, page 467
- How Does TDV Caching Work?, page 474
- TDV-Created Caching Objects, page 479
- What Incremental Caching Options Are Available in TDV?, page 481
- About Native and Parallel (Bulk) Caching, page 482
- Cache Status and Events Reporting, page 483

The following illustration provides a simplified representation of caching with TDV.



Key Features and Benefits of Caching Data

Many business applications are being developed and deployed on multi-tier environments involving browser-based clients, web application servers, and databases. These applications require data on demand. Caching is an effective way to achieve high scalability and performance. Caching provides:

- Reduced impact on operational data stores.
- Redirection of reporting and analytic queries to cache data stores, eliminating their impact on the performance of OLTP systems.
- Improved performance of TDV requests.
- Materialization of intermediate results (expensive to compute) boosting the performance of requests that depend on them.
- More flexibility.
- The opportunity to move cached data result sets to systems that provide optimal processing and analysis.

TDV Caching Concepts

This section introduces many of the concepts associated with TDV caching. Becoming familiar with these concepts can help you determine if you want to use caching and which TDV caching options might work best for you.

What Is a Cache?

A *cache* is a location that holds a copy of frequently accessed data that is otherwise expensive to retrieve or compute.

Why Use Caching?

Data caching is an important data design option that provides a valuable way to replicate data or store data closer to where it is consumed, to enhance application performance. Caching can also reduce the impact of resource-intensive queries being run against database tables by distributing the workload over several storage locations. Instead of running the same query multiple times to return the same result set, you can use a cache to store the results, and then reuse them.

When Should You Configure Caching?

If direct queries to the database are resulting in performance that is causing noticeable delays in data retrieval, it might be time to consider data caching.

What Types of Caching Are Available?

TDV has designed several caching options so you can create caches that meet your needs. Among these options are:

- Full refresh caching—All data in the cache is retrieved when a cache refresh action is initiated.
- Incremental refresh caching—Only data that has changed is retrieved and updated in the cache when a cache refresh action is initiated.
  - — Pull-based incremental caching—The incremental cache refresh action is initiated by a request from the consuming client application. This method could be considered on-demand.
  - — Push-based incremental caching—The incremental cache refresh action is automatically initiated as defined during configuration of the option. The data in the cache is refreshed regardless.

Additional enhancements, available through your database vendor, worth considering for caching include:

- Native loading features such as FastLoad
- Parallel caching feature that partitions the cache data and then runs parallel processes to load the data into the cache target

What is a Full Refresh Versus an Incremental Refresh?

You can also have your cache data refresh happen incrementally or in full. A full refresh would retrieve the data in the cache in its entirety, include data that changed since the last cache refresh and any data that is unchanged. With an incremental cache update, only data that is changed (newly added, modified, and deleted) is updated in the cache when the refresh action runs.

On the Caching tab under the Advanced options, you can select one or the other for your cache. If you require a push-based incremental refresh, there are several special required configuration steps that are explained in Push-Based Incremental Caching, page 629.

What Are the Incremental Caching Options?

TDV provides for two methods of incremental caching. The updates to the cache can be propagated using one of the following two modes:

- Pull-Based Incremental Caching

  Pull-based incremental caching can be set up using the TDV caching mechanism with two scripts. With pull-based incremental caching the user or client application must make a request for their copy of the cache to be synchronized with the centralized cache copy. This asynchronous method or cache data update can improve cache loading times because after the initial data is loaded into the cache each subsequent load is only updating the cache with the smaller data changes. For more information on pull-based incremental caching, see Setting Up Pull-Based Incremental Cache, page 531.

- Push-Based Incremental Caching

  Push-based incremental caching can be set up using the native TDV caching mechanism, plus the TDV Change Management Service (CMS), to configure a Central Event Server. This synchronous method of cache data update enforces cache consistency across all copies of the cache data. To protect the data in the incremental cache, you can configure the TDV server to back up the primary Central Event Server. To keep transactional semantics from differing between the sources and the caches, changes in the source are captured and applied to all affected views. For more information on push-based incremental caching, see Setting Up Push-Based Incremental Caching for Oracle, page 536.

What TDV Resources Can I Cache?

Caching lets you store data generated from a view, table, or procedure from across data sources. Table objects include tables, views from data sources, and TDV views. Procedure objects include SQL scripts, parameterized queries, and XML transforms.

What Cache Storage Options Are Available?

A storage option determines the location and some of the behavior associated with a cache. A cache can be stored as a:

- Default—Cache data is stored in a TDV determined database location.
- Automatic—Cache data is stored in a TDV determined structured file location.
- Single Table—Cache data is stored in a user determined structured database location.
- Multi-Table—Cache data is stored in a user determined database location in a user determined number of caching tables.

  For single table caching, several snapshots (a picture of your data taken at a point in time) of data is stored in the same physical table. Because several snapshots are stored in the same table, index on the cached data is less effective. With the multi-table cache option, TDV uses several physical tables for each cached resource so that each snapshot is stored in its own table. This feature can increase the speed of deleting old cache data and indexing the newly cached data. Multi-table caching is primarily meant for caches that contain a substantial amount of data which is retained for a long time. For details about setting up this type of caching, see Creating a Multiple Table Cache on a Database Target, page 501.

What Is a Cache Target?

A cache target is a file, database table, or multiple database tables used to store data for a specified period of time.

For a data source to be a cache target, it must conform to the prerequisites specified in Cache Requirements and Limitations, page 483. While it is possible to have the same cache resource act as both source and target, it is more common to have them be different. Data from multiple resources cannot share cache target tables.

Consider what cache target is most suitable for the type of data to cache and how you want to interact with it. For example, a cache target might not support important data types in the source. Review the Cache Requirements and Limitations, page 483. Also check the Cache Data Type Mapping section of the *TDV Reference Guide*.

What Refresh Options Are Available for a Cache?

TDV cache data refresh is highly configurable. You can use the standard scheduling tools available through the TDV Studio UI, or you can define custom refresh behavior. The following standard refresh options are available:

- According to a policy—Reusable cache policies can be used to define cache refresh schedules. The policies can be used to define the refresh options for tables, views, and procedures.
- Immediately—Refreshes the cache immediately when a TDV user opens the Caching tab and clicks Refresh Now.
- One Time Only—Allows you to schedule a specific date and time for the cache to be refreshed once.
- Periodic—Allows you to define a regular recurring cache refresh action.
- Custom—You can also define SQL scripts, Java programs, or web services that control and define cache refresh behavior. For example, you could write a custom Java program that would initiate a cache refresh if an email was sent to a particular email address with a well formatted subject line. For more information on defining custom refresh behavior, see Refreshing and Clearing Your Cache Programmatically, page 524.

Modifying the query of the view (or table) that you have caching defined on does not automatically update the cache. You must execute a refresh or wait for the next scheduled refresh cycle to get the correct result set.

What is a Cache Policy?

Cache policies define cache refresh and expiration schedules. After a cache policy is defined, it can be used to control the refresh and expiration schedules of Studio resources that you want to cache.

Because the same refresh and expiration schedules can be set for resources in your cache, data consistency and accuracy can be improved for the applications accessing the cached data. For example, if you have an application that relies on cached data for the ORDERS table and the ORDERDETAILS table, you can use a cache policy to make sure that their caches are refreshed at the same time.

Cache policies also enhance cache performance. With cache policies TDV can calculate the optimal cache load strategy for performance because the list of resources that are part of the cache refresh action for a given policy are well known.

With cache policies, cached data consistency is all or nothing. If one of the resources included in the cache policy fails during refresh action, all of the caches are rolled back to the previous successful snapshot of data. For this reason, cache policies and incremental caching are not supported for use together.

When Does Cache Data Expire and Get Removed From the Cache?

A cache refresh is used to create or update the cache contents. Depending on how you have configured your cache determines how the data from a previous refresh expires and potentially gets removed from the cache.

Expired cache versions are not deleted until all transaction holds using that version are completed or rolled back.

Typical expiration schedules are:

- Never—Keep the current and all previous data snapshots in the cache.
- Periodic—Allows you to define a regular recurring cache expiration time. For example, after a specified number of weeks or years.
- According to Custom Criteria—You can control cache data refresh and clearing by defining custom SQL scripts, Java programs, or web services.

You can also control whether the cache data clears:

- Immediately—Clear the cache using the **Clear Now** button. This marks the old cached version of the data as expired so that new requests cannot use it; then the system initiates a background task to clear the expired cache version when it is released.
- On Failure—Selecting this option would clear the cache if a refresh fails. This option allows access to previously cached data during a refresh.
- Before the Refresh—Selecting this option would automatically clear the cache before starting a refresh. Any clients attempting to read from the cached data must wait for the new data. Forces all new requests to use the new cache version, marks the older cached version as expired but not actually cleared, and new requests for the cached resource must use the latest data provided by the cache refresh.
- Periodically At Expiration Time—On cache expiration, if a time is defined in the Expiration Schedule portion of the screen, the old cache data is cleared out.

- According to Custom Criteria—You can control cache data refresh and clearing by defining custom SQL scripts, Java programs, or web services.

What is Cache Versioning, Transaction Integrity, and When Are Old Cache Contents Cleared Out?

TDV cache versioning preserves transaction integrity by maintaining old cache versions until existing transactions are completed. Expired cache versions are not deleted until all transaction holds using that version are completed or rolled back. Even when you manually initiate cache clearing, that action only marks the old cached version as expired so that new requests cannot use it; then the system initiates a background task to clear the expired cache version when it is released.

TDV can be configured to gracefully handle long-running cache refreshes by retaining all previous cache data snapshots, which preserves usability of the data while a cache is actively being refreshed in the background.

To preserve read consistency, long-running queries that use data from a cached view are allowed to complete computations with the original cached view that it was using at the time the cache refresh started.

Can Caches Be Exported and Imported?

All of the cache information that you define within TDV can be exported and reimported into TDV. Using the TDV export functionality can be an effective way to manage your TDV development work. If done routinely, it can provide a valuable rollback snapshot of your work.

How Does Error Handling Work for Resources that use Cache Policies?

If a cache refresh is initiated by a cache policy and it fails at any point before completion, all the caches are rolled back to the last successful snapshot.

How Does Error Handling Work for Resources with Individually Defined Cache Refresh and Expiration Schedules?

If the data sources providing the data to be cached or the data source used to store the cached data fail during the refresh, the cache is marked as failed or stale depending on whether the previous snapshot is available.

If the TDV server fails during the refresh the cache is marked as failed or stale when the server is restarted.

For caches that perform a full refresh, if the server dies in the middle of a refresh, the refresh attempt is marked as Failed on server restart. Any partial data is eventually processed by the garbage collector.

For caches that perform incremental refreshes, if the server dies in the middle of a refresh, the cache status is marked as DOWN when the server is restarted.

How Does TDV Caching Work?

TDV caching options and how caching works for the different TDV resources are worth understanding before choosing one over the other. This section covers the following topics:

How Does Table and View Caching Work?

Within TDV, it is possible to configure caching for views and tables. A SELECT * is run against the view or tables and every row returned is placed into the cache. Because of this:

- The cache will be as large as the output set of the table or view.
- All requests to that view or object come from the cache after its first load.

You can create a view based on the data or object that you want to cache. You could then later limit the data that is included in the cached view for performance reasons.

TDV caching is best used for materialized views under any of these conditions:

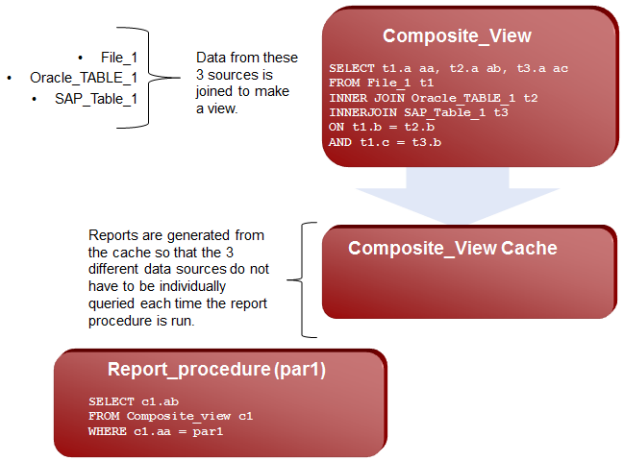
- A view requires substantial execution time.
- The data does not change significantly within a given period.
- The data source should be protected from excessive usage.

For view caches, if a query references a cached view, and the cache is not loaded, it starts a refresh and the query is blocked until the cache is loaded.

In the following example, Sales_Table is an Oracle table, and the UK_Sales_View is a TDV view of that table. If you cache at the table level, all 10 million rows and 15 columns are selected from Sales_Table and put into the cache. However, if you cache the *view*, the view projects only three columns and has a WHERE clause that limits the result set to 100,000 rows.

The WHERE clause and the SQL in the UK_Sales_View are pushed to the Oracle database for processing. Only 100,000 rows and three columns are extracted from Oracle, sent over the network, and placed in the cache. This is 500 times less data for the same result. The cache is smaller, the refresh is quicker, and there are fewer loads on both the Oracle database and the network during the refresh.

In the following example, Composite_view joins a file, an Oracle table, and an SAP table. Report_proc uses Composite_view and pushes a predicate against it. Because File_1 does not support pushes, the predicate requires a full scan of File_1 and SAP_Table_1 each time, slowing performance with the joins and the fetches. If Composite_view is cached, the join is performed only once.



What Is Procedure Caching?

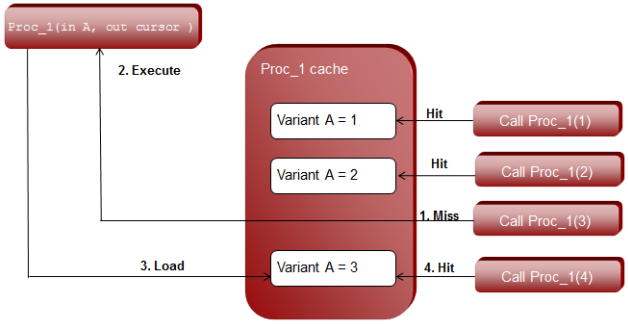
Tabular results from procedure-based resources can be cached by TDV. Procedures with one or more input parameters can have many variants (unique sets of input parameters) and return different result sets based on those input variants. More than one result set from the same procedure can also be cached.

To track variants, the input parameter values are converted to a string to compare for uniqueness. If this string exceeds 255 characters, the procedure call is not cached. Instead, it is executed as if caching was disabled.

For example, Proc_1 has one input parameter (A, an integer) and returns a result set. Proc_1 can be called for different values of A. The caching mechanism executes the procedure and stores the result set for each unique set of input parameters (in this case, integer A). If two variants are in the cache, one with A= 1 and another with A = 2, and Proc_1 is called with A equal to 1 or 2, there is a hit on the variant and the corresponding cached result set is returned.

If Proc_1 is called with a variant that is not currently cached, like A = 3, a "cache miss" occurs, and the procedure is executed with A = 3. The result set is directly returned to the caller, and both the variant and the result set are written to the cache. If proc_1 (3) is called again prior to expiration of the result set, where A = 3, the cached result set is returned immediately.

Because there can be many variants, it might not be possible to store them all. By default, the maximum number of stored variants is 32, but you can change the value in the Maximum number of procedure variants field in the Advanced section of the Caching panel. If the cache is full, the least recently used variant is swapped out for the latest variant.



The following procedural resources can be cached:

- Java procedures
- Packaged query procedures
- Parameterized SQL procedures
- Physical stored procedures that are introspected
- SQL script procedures
- Transformation procedures—basic, streaming, XSLT, and XQuery
- Web service operations

The procedure caching process uses one storage table for each output cursor and an additional storage table for any scalar outputs. For example, a procedure with two INTEGER outputs and two CURSOR outputs would use three tables:

- One for the pair of scalars
- One for the first cursor
- One for the second cursor

If a procedure has input parameters, the cached results are tracked separately for each unique set of input values. Each unique set of input parameter values is called a variant.

For example, a procedure with an INTEGER input parameter would have separate results cached for inputs of 1, 3, and 5, or whatever value. A procedure with two INTEGER input parameters would have separate results cached for inputs (1,1), (1,2), and (x,y).

Note: A procedure cache that uses non-null input parameters must be seeded with at least one variant from a client application other than Studio, for the Cache Status to change from NOT LOADED to UP. Using the Refresh Now button does not change the status of a cache that is not loaded. Even if procedure caching configuration is correct, the status does not show that it is loaded until a client seeds the cache.

When a procedure cache is refreshed, all the cached variants already in the table are refreshed. If no variants have yet to be cached, then nothing is refreshed or only the null input variant is refreshed. You can refresh a procedure cache from Studio or using the RefreshResourceCache procedure. (See the *TDV Application Programming Interfaces Guide*.)

What Is Transaction Result Caching for Procedures?

Transactional result caching is available for procedures. For each unique set of input parameter values, the caching data is collected one time. When transaction result caching is enabled, results are captured in memory the first time the procedure is run during any transaction; additional calls to the procedure with the same input parameters return the cached data. Transaction result caching has no refresh, clear, status, or tracking available.

Transaction result caching is useful in the following scenarios:

- The TDV procedure is run against every row in a query.
- The SQL Script logic inherently requires enough processing time to make the entire query's cumulative execution time unacceptable.
- The results from the TDV procedure performs lookups against a table whose data changes frequently.

Results of calls to the SQL script are stored in memory, and later calls with the same parameters return data from the cache if the data exists there, or execute the SQL script and store that variant in memory.

In Transaction Result Caching:

- Data cached for the transaction persists only for the duration of the transaction.

- Transactional cached information is stored in memory.

For example, consider the following two TDV resources. Each call to lookup requires a SQL lookup to a table. If the composite view returns 1,000,000 rows without Transaction Result Caching, there are 1,000,000 database queries performed by lookup. If transaction processing is enabled and field1 does not contain unique data, each result and input combination is stored temporarily in memory. When a duplicate field1 value is returned, TDV returns the previous result value in memory for that input. All the values are discarded when the transaction ends.

Composite View	Lookup SQL Script
`SELECT lookup(field1) result, field2, field3 FROM largeTable`	`PROCEDURE lookup(IN iVal INTEGER, OUT resVal INTEGER) BEGIN DECLARE c CURSOR; OPEN c AS SELECT f1 FROM largeSlowLookupTable WHERE lookupVal = iVal; FETCH c INTO resVal; CLOSE c; END;`

To enable transaction result caching, check the "Execute only once per transaction for each unique set of input values" check box in the Transaction Options section of the Info tab for any procedure.

How Does Caching Data in a Cluster Environment Work?

Running TDV server in a cluster environment, whether the TDV Active Cluster or any other clustered environment, you can cache data.

When the TDV server is running in a cluster and the cache is in Default Cache or Automatic mode, each server keeps a separate copy of cached data on its local file or database system, and no sharing of data is performed.

When the server is running in a cluster and the cache is in single table or multi-table mode with an identified data target, all servers in the cluster access the same data target for cache information. In addition, the servers cooperate to ensure that a minimum number of refreshes occur. For example, if two servers both need to refresh the data in a cache, only one of them performs the refresh, and then both servers use the updated data.

Note: The use of the File-Cache data source or other file-based data sources is not recommended when in a cluster. The use of network-mounted files is also not recommended because the file data sources do not support file locking.

In a cluster environment, any attempt to refresh a cache uses the TDV cache status table to determine if any other refreshes are in progress. If no others are in progress, the refresh starts and all other servers are notified to reread the TDV cache status table (for information on what the cache status table is, see TDV-Created Caching Objects, page 479). If a refresh is already in progress, the server contacts the server that is currently refreshing and waits for that refresh to complete.

In a cluster environment, an attempt to clear a cache marks the cache data for clearing in the TDV cache status table. The background task to actually delete the data contacts other cluster members to determine what cache keys are no longer in use so the task can safely remove only rows that no server in the cluster is accessing. After the clear task completes, all other servers are notified to reread the TDV cache status table.

A scheduled refresh in a cluster relies on the trigger system cluster feature to ensure that triggers are processed only the appropriate number of times. File cache schedules are configured to fire separately on each server. Database cache schedules are configured to fire only once per cluster. You can create trigger resources, instead of using the schedule options provided on the Caching panel, to control this behavior. (See Triggers, page 391.)

TDV-Created Caching Objects

This section provides an overview of some of the key objects created to facilitate TDV caching. Depending on how you set up your cache, the following objects are created either by TDV or by you. If you decide to use the automatic storage type selection, the names of the objects are set by TDV. If you choose any of the other options you can name the tables anything you want. TDV suggests using names that are easy for you to identify as cache data, cache tracking, and cache status.

Cache Status

The status table (or file) is used to track which views and procedures currently have cached data stored, when they were last refreshed, and so on.

Each Studio data source used as a caching target must have its own status table.

Cache Tracking

The tracking table (or file) is used to track the tables, views, and procedures that are using a particular cache target. It is also used to track what tables in the cache target are in use.

Each Studio data source used as a caching target must have its own tracking table.

Cache Data

The name of the object you want to store cached data for is appended with cache. These are the file or database tables that will hold the cached data. Resources cannot share the same data cache.

The cachekey Column

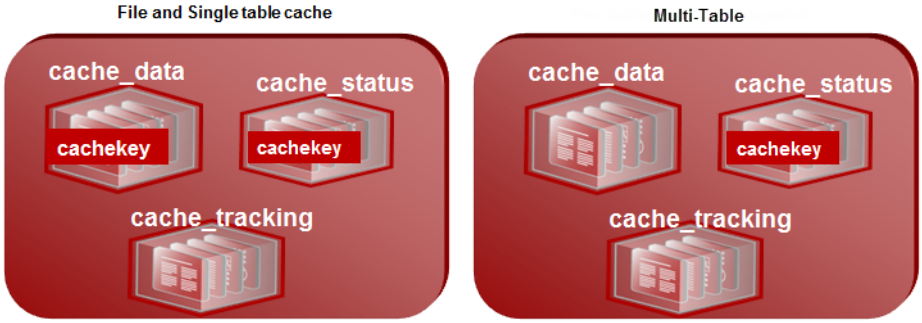
The cachekey column is a way to uniquely identify a particular snapshot of cached data. TDV uses it to determine what data in the cache to return for a query request from an application, to determine cache versioning, transaction integrity, and when old cache records are ready to be cleared out. If you decide to use some of the more advanced caching options, you can use the cache key in your custom scripts to drive caching actions such as syncing cache data to a cache copy on request.

The cache data table, for most caching options, has a cachekey column that contains a unique integer identifier. The cache status table identifies the cachekey value that is associated with current data or with specific parameter input variants for procedures.

The cache key is a unique number generated for each cached result set. For most caching configurations, it is the only caching table column automatically generated by the TDV Server.

Because the cache key is a unique integer for each snapshot of data, it can be useful for indexing, when defining pre- and post-refresh action scripts, or when defining scripts for incremental caching. Because the cache key is used often to filter data, it should be the first column of the index when indexing the cache table.

For multi-table caching, the cachekey column is not in the target table, but it is in the cache status table to support leader election, garbage collection, cluster communication, and for consistent cache status.



The cache metadata tables contain some abbreviated values. For example:

Metadata Values	Meaning	Description
A	active	Cache is loaded.
I	in progress	Cache is currently refreshing.
F	failed	The loading of the cache failed for some reason.
C	cleared	Cache has been cleared.
K	key	A special value used to indicate the key row in the table.

What Incremental Caching Options Are Available in TDV?

After initial cache loading, incremental caching improves caching performance by only *updating* the cache, rather than *reloading* it, to reflect changes in the data source. Resources are incrementally maintained as highly available and responsive caches that are kept in sync with real-time data in the data source. TDV's incrementally maintained caches provide the authenticated and authorized end-user client with synchronized data services ready with data results on demand. The following two options are available:

- Pull-Based Incremental Caching, page 481
- Push-Based Incremental Caching, page 481

Pull-Based Incremental Caching

Pull-based incremental caching can be set up using the TDV caching mechanism with two scripts. With pull-based incremental caching the user or client application must make a request for their copy of the cache to be synchronized with the centralized cache copy. For more information on pull-based incremental caching, see Setting Up Pull-Based Incremental Cache, page 531.

Push-Based Incremental Caching

Push-based incremental caching can be set up using the native TDV caching mechanism, plus the TDV Cache Management Service (CMS), to configure a Central Event Server. If you have subscribed to notices from objects that store data in an incremental cache, subscribed clients are notified whenever the result sets are updated. For more information on push-based incremental caching, see Setting Up Push-Based Incremental Caching for Oracle, page 536.

About Native and Parallel (Bulk) Caching

Native cache loading is designed to enhance caching performance loading the cache. The Enable Cache Loading and Enable Parallel Loading Studio configuration parameter controls whether native caching is enabled or disabled. By default, the native cache loading is enabled.

When a single-source, pass-through view is cached, native cache loading might be supported.

There are several ways to tune the performance associated with data caching. If you have set up caching performance configurations for native and parallel caching options, TDV determines which option is ideal.

If native and parallel caching options cannot be used to move the data, TDV attempts to use JDBC to do the cache loading.

The following cache loading options are supported by TDV:

- INSERT/SELECT statements
- Bulk load with the LOAD utility
- Bulk load with bcp, utility (bcp.exe)
- Bulk load with external tables
- Bulk load through database links
- Bulk load using the Bulk Load utility
- Parallel cache load

When the native caching option can be performed within the same data source, that is the source of the cache data and the target for the cache data reside in the same database or schema, and the data source supports INSERT/SELECT statement, the cache loading will use INSERT/SELECT statement. With the INSERT/SELECT statement method of native caching, almost all data types are supported.

If the INSERT/SELECT statement cannot be used for native caching, TDV tries to use the mechanism that the target data source supports. These options vary by database type.

Target Database Type	Bulk Load Facility	Notes
DB2	LOAD utility	
Microsoft SQL Server	bcp.exe utility	The bcp.exe path must be specified in TDV.

Target Database Type	Bulk Load Facility	Notes
Netezza	external tables	Additionally, if DISTRIBUTE can be used to get more performance gains through parallel processing, it is used.
Oracle	Database Links	Requires a database link configuration in Oracle database and a link name in the definition of the TDV cache target data source.
PostgreSQL including Greenplum	COPY command	
Sybase IQ	LOAD_TABLE statement	Requires configuration of iAnywhere and DSN.
Teradata	FastExport and FastImport	
Vertica	Bulk Load utility	

Cache Status and Events Reporting

Cache status is reported in Studio on the Caching panel for the resource, and on the Cached Resources console in the Manager.

The server event log is set to log all cache events by default. These events can be seen on the Events console in the Manager. They can also be found in the events log files and in the SYS_EVENTS system table. The logging levels for these events can be set in the Manager Configuration panel (under TDV Server > Events and Logging), as with all other supported events.

Cache Requirements and Limitations

Data caching has many important requirements and limitations to consider. This section describes:

- Supported Data Sources for Caching, page 484
- Supported Cache Target Storage Types, page 484
- Privileges and Caching, page 487
- Caching Limitations, page 488

Supported Data Sources for Caching

TDV caching supports most of the data sources (including adapters) supported by the TDV Server. For the current list of supported data sources in TDV, see the *TDV Installation Guide* or *TDV Installation and Upgrade Guide* for your version of TDV.

Supported Cache Target Storage Types

Your cached data can be stored in database tables or as a file. Database caching lets you store result sets in a database so that further manipulations of the result set can be performed in dramatically less time. If any of the cached results are to be used by other views, or be filtered or sorted, then the cache should be stored in a database.

Data from one resource can be cached into one or more cache targets, but multiple resources cannot use the same cache target.

Cached resources are best stored in tabular form. The schema of the resource data must match the schema of the selected data source table. The cache database can have more columns than the source, but the data types of the columns must match. DDL can be generated to create a caching table that matches your view or procedure output.

File storage options are ideal when the results are not sorted or filtered for use in other views. File caches do not store index information, nor do they allow you to push SQL logical operators and filters to the data source. Because there is no index information, any selection of a subset of a view requires loading the cached view into memory for a row-by-row scan. SQL operations are executed on the cached data within TDV memory.

Note: If you build additional queries on large file cache views, significant memory will be required for a full table scan.

TDV supports the following as cache targets:

Cache Target	TDV Support	Parallel Cache Target Support	Native Cache Target Support	Notes
File	Active	Active		Typically best for demonstrations or caching of a few hundred rows.
Greenplum 4.1	Active	Active	Active	
Greenplum 4.3	Active	Active	Active	
HSQLDB 2.2.9	Active	Active		
IBM DB2 LUW v10.5	Active	Active	Active	Native load with insert and select, and DB2 Load are supported.
Microsoft SQL Server 2008	Active	Active	Active	The DBO schema must be selected and introspected as a resource prior to attempting to cache data.
Microsoft SQL Server 2012	Active	Active	Active	The DBO schema must be selected and introspected as a resource prior to attempting to cache data.
Microsoft SQL Server 2014	Active	Active	Active	The DBO schema must be selected and introspected as a resource prior to attempting to cache data.
Microsoft SQL Server 2016	Active	Active	Active	The DBO schema must be selected and introspected as a resource prior to attempting to cache data.
MySQL 5.1	Active	Active	Active	
MySQL 5.5	Active	Active	Active	

Cache Target	TDV Support	Parallel Cache Target Support	Native Cache Target Support	Notes
Netezza 6.0	Active	Active	Active	Native load with insert and select is supported. Parallel cache processing is achieved using the native DISTRIBUTE syntax. Procedure caching is supported.
Netezza 7.0	Active	Active	Active	Native load with insert and select is supported. Parallel cache processing is achieved using the native DISTRIBUTE syntax. Procedure caching is supported.
Oracle 10g	Supported			Native load with INSERT and SELECT is supported. Native load with DB link is not supported.
Oracle 11g and 11g R2	Active	Active	Active	
Oracle 12c	Active	Active	Active	
PostgreSQL 9.1	Active	Active	Active	Bulk load is supported. Native loading is supported when the source and target are the same database. If not then Parallel loading is used.
PostgreSQL 9.2.3	Active	Active	Active	Bulk load is supported. Native loading is supported when the source and target are the same database. If not then Parallel loading is used.
SAP HANA SPS 09	Active	Active		

Cache Target	TDV Support	Parallel Cache Target Support	Native Cache Target Support	Notes
Sybase ASE 12.5	Active			
Sybase ASE 15.5	Active			
Sybase IQ 15.2	Active		Active	
Teradata 13	Active		Active	Supported, but with limitations.
Teradata 13.10	Active		Active	Supported, but with limitations.
Teradata 14.10	Active		Active	Supported, but with limitations. Might require Teradata 15 driver.
Teradata 15	Active		Active	Choose tables For Caching is not supported.
Vertica 6.1	Active	Active	Active	Supports the use of native load and parallel cache load together. Native load with INSERT AND SELECT is supported.

Not all caching tables can support data types from all other supported data tables, and so materialized views with certain data types are not compatible for caching at that database. When a data type mismatch occurs, a warning is issued to the developer to prevent further problems.

Not all data sources that could be used for cache storage can support the generation or execution of DDL statements. In some cases, DDL can be generated and presented, but use of an external tool might be required to create the table metadata. In other cases, the DDL is not shown, because it is not applicable.

Privileges and Caching

Caching is defined in TDV on specific resources, you must have access to a resource to create a cache for it. The user who owns the resource being cached is the user identity that cache refresh and clear operations are run with.

TDV caching typically requires the creation of objects on the data source that is acting as the cache target. To create and delete the data and objects necessary for caching with TDV you must have the necessary permissions set up for you in the cache target. Performing a cache refresh or clear requires the READ, SELECT, INSERT, UPDATE, and DELETE privileges on the status table, tracking table, and the data tables. For example, if you use an Oracle database as your caching target, the TDV user that is defining the cache must have privileges on the Oracle database to READ, SELECT, INSERT, UPDATE, and DELETE data and objects directly in the database. Those privileges can be granted by explicit assignment, or by membership in a group with those privileges (implicit assignment).

For someone to use the view or procedure that is being cached to the cache target, they must also have privileges granted to access the cache status and data tables. For someone to read from the cache, they must be granted the SELECT privilege on these tables and READ privileges on folders above these tables.

Privileges are managed automatically when using the Automatic mode of caching.

Caching Limitations

TDV can cache data from many different sources to many different cache targets. Occasionally, there are subtle limitations when dealing with so many systems. For data caching, those limitations are discussed here.

- The view being cached must NOT have a column called cachekey.
- If the cache target does not support a data type available in the source, then any attempt to store the unsupported data type in the cache results in an error. However, it may be possible to cast the unsupported data type to a supported one (for example, cast a BOOLEAN to an INTEGER type) before sending it to the cache target.
- When caching to a table, ARRAY data types are not supported.
- The following resources cannot be cached without being wrapped in a view or procedure:
  - Procedures with no outputs; that is, no data to cache.
  - Procedures with input parameters that require the use of cache policies.
  - XML files that have been introspected.
  - System tables.
  - Non-data sources such as folders, and definition sets.

Incremental Caching Limitation

Cache policies cannot be used with incremental caching.

Multi-Table Caching Limitation

Multi-table caching is available for tables and views, not for procedures.

Microsoft SQL Server Caching Limitations

When using Microsoft SQL Server with the bcp utility, you might get data type mismatch errors for BLOB or CLOB data types.

For SQL Server data sources, the DBO schema must be selected and introspected as a resource prior to attempting to cache data.

Temporary data can build up in buffer files that are located in the <TDV_install_dir>/cacheloading/sqlserver folder. If there are no cache refresh processes running, it is safe to delete the contents of this folder.

MySQL Caching Limitations

MySQL sets names to lowercase if it is running on Windows, even if you enclose the mixed case string in double quotes. When using MySQL tables as cache targets, either use the TDV browse option to choose the table, or make sure that you assign lowercase names to tables.

The TDV native load option cannot load binary data.

When using MySQL as a cache target, note the data type limitations, by data source, listed in the following table. Also when MySQL 5.5 is the cache target, time and time stamp data types lose precision for milliseconds and other fractional seconds.

Cache Data Source	Data Types Not Supported	Cache Target
DB2 9.5	BLOB	MySQL 5.5
Oracle 11g	BLOB LONGRAW	MySQL 5.5
Sybase 15	BINARY, IMAGE, VARBINARY, or TIMESTAMP	MySQL 5.5

Cache Data Source	Data Types Not Supported	Cache Target
SQL Server 2008	BINARY, IMAGE, VARBINARY, or TIMESTAMP	MySQL 5.5
SQL Server 2012	IMAGE	MySQL 5.1

Oracle Caching Limitations

If the same Oracle database instance is acting as the source of and target for your cached data:

- LONG and LONG RAW data types. SQL*Plus is unable to SELECT a LONG RAW column. To work around the issue, you can convert the LONG RAW data types to BLOB.
- INTERVAL DAY TO SECOND and INTERVAL YEAR TO MONTH lose precision when they are cached using INSERT and SELECT statements.

SAP HANA Caching Characteristics

Cache target tables in SAP HANA are created as column-store tables, rather than the row-store tables more commonly used in relational databases. If partitions are used, SAP HANA uses round-robin partitioning to equally distribute rows to partitions. The table used for caching does not have to have primary keys.

Teradata Caching Limitations

Teradata has a known issue that affects data caching for TDV. Because of this issue, you might be unable to cache data, and you might get incorrect query results against Teradata when Ignore Trailing Spaces is set to FALSE in TDV. The issue is caused by the Teradata driver's management of character data when using UTF-8 character sets.

To solve both the caching and the query problems, you can do one of the following:

- Change the global server setting for Ignore Trailing Spaces to true. (In Studio, choose Administration > Configuration. Locate and select Ignore Trailing Spaces, and click True for Value.)
- Change the Teradata connection string to use UTF-16 by substituting CHARSET=UTF16 for CHARSET=UTF8, and save the data source. Then recreate the cache_status table and refresh the cache.
- Change the Teradata connection string to use ASCII by substituting CHARSET=ASCII for CHARSET=UTF8, and save the data source. Then

recreate the cache_status table and refresh the cache. This solution does not work for data that contains multi-byte international characters because the characters are not saved or retrieved correctly.

To solve just the caching problem, cache data in a different data source (instead of Teradata).

To solve just the query problem, provide query hints (see Specifying Query Hints, page 228) on queries against Teradata where filters are on CHAR columns:

```
{ OPTION IGNORE_TRAILING_SPACES="True" }
```

Teradata Multi-Table Caching Limitations

The Teradata Fast Export and Fast Load features are supported for caching with the TDV multi-table caching option. The cache must have no duplicate rows of data and be configured as specified in Configuring Teradata for Use as a Multi-Table Cache Target, page 502.

- Teradata FastLoad requires that the target table be empty.
- Teradata has limitation on how many concurrent FastLoad and FastExport tasks can run in parallel. Parallelism is controlled by the MaxLoadTasks and MaxLoadAWT parameters. Any FastLoad task exceeding the limitation is rejected.
- For Teradata, the maximum session for each FastLoad or FastExport job is limited to the number of AMPs of the Teradata database. Typically, eight sessions work well for most scenarios.
- For Teradata, a row fetch size bigger than 64 KB causes a Teradata error. Teradata big objects can be configured using Teradata to return data in differed transfer mode. Refer to your Teradata documentation to determine the best solution for you if you have data rows that return 64 KB or greater of data.
- The following data type and function support restrictions exist.

Data Source	Cache Target	Data Types Not Supported	Functions Not Supported
Oracle	Teradata	BLOB, CLOB, LONG, LONGRAW, NCLOB	No results returned after refreshing the cache against INTERVALDAYTOSECOND and INTERVALYEARTOMONTH.
SQL Server 2008	Teradata	BINARY, IMAGE, NTEXT, TEXT, VARBINARY	

Data Source	Cache Target	Data Types Not Supported	Functions Not Supported
Sybase	Teradata	BINARY, IMAGE, TEXT, VARBINARY	
Teradata	Teradata	BYTE, BLOB, CLOB, LONGVARCHAR	
Vertica 5.0 and 6.1	Teradata	BINARY, VARBINARY	

Vertica Caching Limitations

Because of Vertica length limits, mapping of any data type (BINARY, CHAR, VARCHAR, BLOB, and so on) to Vertica cache with length greater than 65000 results in an error, regardless of the data source.

Similarly, Vertica only supports precision up to 15 digits. Any data that you have past 16 digits will get rounded by Vertica. This precision limitation is particularly noticeable when working with REAL, FLOAT, and DOUBLE data types.

Native (Bulk) Caching Limitations

Depending on your cache target, there are a few limitations to be aware of when trying to use this performance option.

DB2 Native (Bulk) Loading Caching Limitations

When bulk load is in use, the following data types cannot be cached to DB2:

Data Source	Data Types Not Supported	Cache Target
Oracle	BLOB and LONGRAW	DB2
SQL Server	BINARY and VARBINARY	DB2
Sybase IQ	BINARY, BLOB, and VARBINARY	DB2

TDV Native Loading Option MySQL Limitation

The TDV native load option cannot load binary data.

TDV Native Loading Option Teradata Limitation

If the TDV native load option is active and the data identified to be moved to the cache has one or more duplicate rows, Teradata 13 will allow the duplicate rows.

TDV Native Loading Option Database Link Limitations

For database links, the following data type and function support restrictions exist.

Data Source	Cache Target	Data Types Not Supported	Notes
DB2	Oracle 11g	REAL, GRAPHIC, LONGVARCHAR, TIME, TIMESTAMP, BLOB, CLOB	CAST, AVG, and SUM functions lose precision
Sybase ASE 15	Oracle 11g	TEXT, IMAGE	AVG, COUNT, MAX, MIN, and SUM functions are not supported.
Oracle 11g	Oracle 11g	BLOB, CLOB AND NCLOB LONGRAW AND LONG	
SQL Server 2008	Oracle 11g	NTEXT, TEXT, DATE, IMAGE Oracle treats empty strings as NULL values.	

Native (Bulk) Caching DDL Creation Limitations

When the data source tries to create a table that contains certain data types, some data types are not supported for particular data source and cache target combinations.

Data Source	Cache Target	Data Type Not Supported
DB2	Netezza 5 or Netezza 6	BLOB, CLOB
Sybase ASE 15	Netezza 5 or Netezza 6	TIMESTAMP, BINARY, IMAGE, TEXT, VARBINARY
SQL Server 2008	Netezza 5 or Netezza 6	VARBINARY, TIMESTAMP, BINARY, TEXT, IMAGE, NTEXT
Oracle	Netezza 5 or Netezza 6	BLOB, CLOB, NCLOB, LONG, LONGRAW

Data Source	Cache Target	Data Type Not Supported
Any data source	Vertica	Any data type (BINARY, CHAR, VARCHAR, BLOB, and so on) with length greater than 65000

Netezza Caching Tips from an Expert

When Netezza is the target of your cache, the experts recommend that you configure the Netezza database Transaction Isolation levels to the default value of Serializable.

Setting Up Caching

This topic describes how to use data caching in TDV to improve performance:

Caching Transaction Results of a Procedure

The transaction result caching feature can be useful as an alternative to full procedure result caching if the primary need is for transaction isolation instead of storing results for repeated access between transactions. When transaction result caching is enabled, results are captured in memory the first time the procedure is run during any transaction; additional calls to the procedure with the same input parameters return the cached data.

For transactional caching, TDV caching does not directly use memory to store the cached result set, but instead persists to disk or a database. Memory-based tables have been used in Oracle and MySQL to achieve even better performance. When the refresh occurs, the object view or procedure is executed and the result set is written to the cache.

To enable transaction result caching

1. From the Studio navigation tree, locate and open the editor for the procedure for which you want to cache transaction results.
2. Select the Info tab.
3. Check the Execute only once per transaction for each unique set of input values.
4. Save your changes.

Caching to the Default Database Target

For the resource being cached, storage depends on the result set output and on retrieval of the entire result set or retrieval of a filtered subset.

For views and tables, the expiration period applies to the whole result set. For procedures, each input variants data has its expiration tracked separately. If used in a cluster environment, each node of the cluster becomes a separate cache view.

If a resource uses a data source that was originally added to TDV Server using the pass-through mode without saving the password, row-based security can affect cache refresh functionality. For details on save password and pass-through login, see Adding a Data Source, page 68.

The default cache is node and fault tolerant.

Limitations

Cache policies cannot be stored in the default cache database.

To enable default caching

1. In Studio, open a view or procedure.
2. Select the Caching tab.
3. Click Create Cache.
4. Under Status, select the Enable check box. If you decide to leave it cleared, you can continue to define the cache, but it will not be active until you select the Enable check box.

When a cache is disabled, all existing cache settings are ignored. The view or procedure is used as if caching did not exist. Toggling between the enabled and disabled state does not cause refreshing of the data or resetting of the expiration date for the data.

5. Save your changes.

   Several fields will display with non-editable information.

   The Data Source, Table Schema (Optional), and Table Prefix fields can be used to determine the location of your cached data within localhost.

6. Set cache policy, cache refresh, expiration, Number of Buckets, Drop and Create indexes on load, and advanced options as necessary.

After data is added to the cache tables by click Refresh Now, you can navigate to the cached data source, execute the view where caching is enabled, and review the cached data. By default, the tables are created on the PostgreSQL database that was defined during the installation of TDV.

Caching to a File Target

If a resource uses a data source that was originally added to TDV Server using the pass-through mode without saving the password, row-based security can affect cache refresh functionality. For details on save password and pass-through login, see Adding a Data Source, page 68.

To enable caching to a file target

1. In Studio, open a view or procedure.
2. Select the Caching tab.
3. Click Create Cache.
4. Under Status, select the Enable check box. If you decide to leave it cleared, you can continue to define the cache, but it will not be active until you select the Enable check box.

   When a cache is disabled, all existing cache settings are ignored. The view or procedure is used as if caching did not exist. Toggling between the enabled

and disabled state does not cause refreshing of the data or resetting of the expiration date for the data.

5. Under Storage, specify Automatic to store the cached data in a system-created file accessible through Studio:
`<HostName>/lib/sources/cacheDataSource`

   The physical files are stored by default in the TDV Server installation directory at: <TDV_install_dir>/tmp/cache

   Each cached resource has a cache table in the cacheDataSource data source. The cache table's name contains the resource type (view or procedure) and a system-generated ID.

6. Save the cache settings.

Pre-Creating Caching Objects for Database Caching

The instructions in this topic are optional and assume that you can create database objects as needed. You will also need to be familiar with the server, schema, and database naming conventions for the database where you are creating these tables. It is suggested that you obtain the documentation for your database of choice. Depending on your corporate policies and procedures, you might have to create the caching objects directly in the database before using Studio to set up caching.

At a minimum, non-file caching requires three tables. One table to hold the cache data, one to hold the cache status, and one to hold the cache tracking information. You can name them anything that is valid according to the constraints of the database where you are creating them. It is suggested that you use names that will be obvious to someone else.

To pre-create caching objects in a database

1. Determine the optimal database for your cache target.
2. Add that database as a data source within Studio, if it is not there already.
3. Determine whether single or multiple cache tables will be needed.
4. Determine how many caches will be needed, or define a process for their request.
5. Locate and open the tool you want to use to create your database tables.
6. For single table caches create the following table using DDL statements similar to the following for each unique TDV resource for which you want to define a cache. The DDL might need to be altered depending on version of TDV your are using, database type where you are creating the table, TDV

resource that you are caching, and whether you want to index the data or not. In the following table, UID stands for user identification. You can name the tables anything you want <resource_metadata> is used to indicate that you will need to specify the names, data types, and other table creation required metadata for the columns you want to create in that table.

Table	DDL Example
<UID_cache>	DROP TABLE IF EXISTS `INV_cache_status`; CREATE TABLE `INV_cache_status` (`cachekey` integer NOT NULL, `resourceid` varchar(255) NOT NULL, `parameters` varchar(255), <table rows>)

7. For multiple table caches, create tables with the following structures for each TDV resource for which you want to define a cache. The DDL might need to be altered depending on version of TDV your are using, database type where you are creating the table, TDV resource that you are caching, and whether you want to index the data or not.

Table	DDL Example
<UID_cache_status>	DROP TABLE "QAN"."mm_cache_status"; CREATE TABLE "QAN"."mm_cache_status" ("clusterid" varchar(40), "serverid" varchar(80) NOT NULL, "resourceid" varchar(255) NOT NULL, "parameters" varchar(255), "status" char(1) NOT NULL, "cachekey" number(10, 0) NOT NULL, "starttime" timestamp(9) NOT NULL, "finishtime" timestamp(9), "cleartime" timestamp(9), "bytes" number(19, 0), "message" varchar(255), "bucket" varchar(255), "incrementalstatus" char(1), "incrementalmaintenancelevel" varchar(255))

Table	DDL Example
<UID_cache_tracking>	DROP TABLE "QAN"."mm_cache_tracking"; CREATE TABLE "QAN"."mm_cache_tracking" ("clusterid" varchar(40), "serverid" varchar(80) NOT NULL, "resourceid" varchar(255) NOT NULL, "catalog" varchar(255), "schema" varchar(255), "table" varchar(255) NOT NULL, "createtime" timestamp(9))
<UID_cache_data>	DROP TABLE "QAN"."mm0"; CREATE TABLE "QAN"."mm0" ("TransactionID" number(10, 0), "TransactionDate" date, <resource_metadata>); CREATE UNIQUE INDEX "mm0_PRIMARY" ON "QAN"."mm0"("TransactionID" asc) DROP TABLE "QAN"."mm1"; CREATE TABLE "QAN"."mm1" ("TransactionID" number(10, 0), "TransactionDate" date, <resource_metadata>); CREATE UNIQUE INDEX "mm1_PRIMARY" ON "QAN"."mm1"("TransactionID" asc) DROP TABLE "QAN"."mm2"; CREATE TABLE "QAN"."mm2" ("TransactionID" number(10, 0), "TransactionDate" date, <resource_metadata>); CREATE UNIQUE INDEX "mm2_PRIMARY" ON "QAN"."mm2"("TransactionID" asc) DROP TABLE "QAN"."mm3"; CREATE TABLE "QAN"."mm3" ("TransactionID" number(10, 0), "TransactionDate" date, <resource_metadata>); ...

8. Re-introspect the data source to make sure that the additional table you have created are available through TDV.
9. Set up your cache and select these tables as the status, tracking and data tables.

Caching to a Single-Table Database Target

For the resource being cached, storage depends on the result set output and on retrieval of the entire result set or retrieval of a filtered subset. Decide on a storage type for caching the execution result. For information on how TDV cached data types are mapped to data source data types, see “Cache Data Type Mapping” in the *TDV Reference Guide*.

If a resource uses a data source that was originally added to TDV Server using pass-through mode without saving the password, row-based security can affect cache refresh functionality. For details on Save Password and Pass-through Login, see Adding a Data Source, page 68.

To set up caching to a single table

1. In Studio, open a view or procedure.
2. Select the Caching tab.
3. Click Create Cache.
4. Under Status, select the Enable check box. If you decide to leave it cleared, you can continue to define the cache, but it will not be active until you select the Enable check box

   When a cache is disabled, all existing cache settings are ignored. The view or procedure is used as if caching did not exist. Toggling between the enabled and disabled state does not cause refreshing of the data or resetting of the expiration date for the data.
5. Under Storage, select Single Table. The cached data is stored in a specified data source. Type or browse to such a storage data source for the cache target.
6. Use Browse to locate and specify the data source where you want to create the cache table. After you select the data source, its full path is displayed in the Data Source field. For data integrity, each resource that is cached should have its own cache data tables.
7. Click Open Data Source. In the Caching section of the Configuration tab, create (execute the DDL), rename or browse to two tables, one for storing cache status data (Status Table) and the other for storing cache tracking data (Tracking Table).
8. Save the data source panels.
9. Navigate back to the view or procedure with the open Caching tab.
10. Click Browse next to the result field in the Table for Caching section. In the Select cache table for result dialog box, specify a location within the data source to create a table for caching. This location can be the root level of the

data source, or a schema. Follow the instructions on the screen to get through the screens that might appear.

For example, when you select a data source, the right side of the screen might say Create Table. When you type a name and click Create, a DDL for result window appears showing the DDL generated by Studio. Click Execute in that window to create the table in the data source. Even though a dialog box might say DDL execution was successful, a Status of CONFIG ERROR plus an Error message may appear on the Caching panel. Click Details for a full description.

11. Save the cache settings. After cache settings are saved, each cached resource appears with a lightning-bolt icon in the Studio resource tree to show that the resource is cached.
12. Optionally, if you would like to have actions happen before or after the cache is refreshed, see Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530.
13. If you want to define pull-based incremental caching for your file-based data cache, see Setting Up Pull-Based Incremental Cache, page 531.
14. If you have data type incompatibilities between your view and your data storage type, see “Cache Data Type Mapping” in the *TDV Reference Guide*.

Creating a Multiple Table Cache on a Database Target

With the multiple table cache option, TDV stores the cache data, from views or tables, in a user specified number of tables. Because the cached data can be stored in different tables, it is possible to:

- Create more effective indexes on the cached data
- Clear old data from the cache faster than with other modes of caching

The multi-table cache option is primarily meant for caches that contain a substantial amount of data which is retained for a long time.

The multi-table feature is designed for multiple cache refreshes. Each refresh populates rows in one table, and because multiple tables are open the performance of caching is improved. Records are not distributed across multiple tables.Multiple tables available for cache refreshes creates a situation where customers accessing data in one of the cached tables does not slow down a refresh action because one of the other cache tables can be used.

If a resource uses a data source that was originally added to TDV Server using the pass-through mode without saving the password, row-based security can affect cache refresh functionality. For details on Save Password and Pass-through Login, see Adding a Data Source, page 68.

Follow the instructions in these sections to use the multi-table cache feature:

Configuring Teradata for Use as a Multi-Table Cache Target

For Teradata, the TDV multi-table option can make use of the Teradata bulk loading FastLoad FastExport functions. You must follow the Teradata documentation for how to set up the FastLoad FastExport features on the Teradata data source and configuring session count. You can then use Studio to complete the configuration of FastLoad FastExport functions.

Note: If you are not using Teradata as a cache target, you can skip to Enabling Caching to Multiple Tables, page 502.

To configure Teradata as the multi-table cache target

1. Open Studio.
2. Select Administration > Configuration.
3. Search for Enable Native Loading and make sure it is set to True.
4. Right-click the data source name in the Studio resource tree and select Open.
5. Select the Configuration tab and under the Connection Information select the Advanced tab.
6. Use the Advanced tab to set values for the following fields.

Field	Description
Enable FastLoad/FastExport for large tables	True means to use Teradata's FastLoad or FastExport utility to speed up query times. Query cardinality is used to decide whether to use Fastpath or JDBC default loading.
FastExport Session Count	The number of FastExport sessions to use for Teradata.
FastLoad Session Count	The number of FastLoad sessions to use for Teradata.

7. Save your settings.

Enabling Caching to Multiple Tables

When clearing a multi-table cache, TDV first attempts to clear the cache using a TRUNCATE command. If the TRUNCATE command is not supported, a DELETE command is attempted.

To enable caching that saves data to multiple tables

1. In Studio, open a view or a table.
2. Select the Caching tab.
3. Click Create Cache.
4. Under Status, select the Enable check box. If you decide to leave it cleared, you can continue to define the cache, but it is not active until you select the Enable check box.

   When a cache is disabled, all existing cache settings are ignored. The view is used as if caching did not exist. Toggling between the enabled and disabled state does not cause refreshing of the data or resetting of the expiration date for the data.
5. Under Storage, specify the storage type as Multi-table. Multi-table is a data caching option that creates one table for each cache refresh up to the cache table threshold.

   This option requires that you specify a database data source to hold the cache tables and that you have sufficient privileges on that data source to CREATE and DELETE cache objects.

   Note: Resources cannot share the same data cache table.
6. Identify a database data source where you have permissions to create and delete objects from the database.

   Note: File data sources are not valid targets for multi-table caching.
7. Use Browse to locate and specify the data source where you want to create the cache table. After you select the data source, its full path is displayed in the Data Source field.
8. Click Open Data Source. In the Caching section of the Configuration tab, create (execute the DDL), rename or browse to two tables, one for storing cache status data (Status Table) and the other for storing cache tracking data (Tracking Table).
9. Save the data source panels.
10. Navigate back to the view or procedure with the open Caching tab.
11. In the Multi-table section, choose between the alternatives shown in the table.

Field	Description
Create Tables Automatically	Provides selections for indicating a table prefix, number of buckets, and index creation for your required cache objects.

Field	Description
Choose Tables for Caching	Provides a mechanism for you to select predefined cache tables. Your predefined cache tables must adhere to the metadata requirements of the cache tables. For data integrity, each resource that is cached should have its own cache data tables.

12. If you selected Create Tables Automatically, you must make choices in the fields listed in the table.

Fields	Description
Table Catalog (Optional)	Depending on how your data source was defined and introspected, you might have one or two levels of folders under the data source in the Studio resource tree. The table catalog refers to the first level of folder under the data source. If your data source contains subfolders, type the name of the first level here. The name is case-sensitive.
Table Schema (Optional)	The table schema refers to the second level of folder under the data source. If your data source contains subfolders, type the name of the secondary level here. The name is case-sensitive.
Table Prefix	Prefix to add to the cache tables and any indexes that you want to create. A unique prefix is required.
Number of Caching Tables	Use to indicate the number of snapshots of the cache to keep. Start with a value of 3 and then determine if that number needs to be increased to accommodate your specific caching needs. The number of cache tables needed depends on how often data is cached, how much data is cached, how often refresh occurs, and the longest running transaction that accesses this cache. For example, if the longest running transaction is 4 hours and refreshes happen every hour, you should define at least 4 cache tables. If a cache is to refresh every 2 hours but some transactions run for 6 hours, configure at least 4 cache tables. Each row in cache status table has the name of the cache table containing the snapshot. A cache table is not eligible for garbage collection as long as there is an active transaction that uses it. TDV attempts to detect cache table sharing, but it is your responsibility to ensure that each cache table is used by only one resource.
Create Cache Tables	Runs the DDL to create the tables needed for caching.

Fields	Description
Drop indexes before load and create indexes after load	When selected, all indexes listed in Indexes tab of the view are part of the generated DDL and are created. Enabling this check box makes TDV drop indexes before loading data and recreates them after, to improve load time.

13. If you want to have the multi-table caching option to drop and create indexes after the cache loads, select the Indexes tab and review or define an index for the view or table. For information on how to create the index using the tab, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.
14. If you selected Choose Tables For Caching, you must make choices in the following fields:

Fields	Description
Add Table	Use to add additional tables for cache storage.
Table Name	Browse to select or create the table for a multi-table cache. You can also type the resource tree path. This table must not be used to store data for other caches.

15. Save the cache settings. After cache settings are saved, each cached resource appears with a lightning-bolt icon in the Studio resource tree to show that the resource is cached.
16. Optionally, if you would like to have actions happen before or after the cache is refreshed, see Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530.
17. Optionally, if you want to define pull-based incremental caching for your file-based data cache, see Setting Up Pull-Based Incremental Cache, page 531.
18. If you have data type incompatibilities between your view and your data storage type, see “Cache Data Type Mapping” in the *TDV Reference Guide*.

Enabling Cache Data Storage to Multiple Data Sources

For each eligible data source, you can specify whether you want the cache status and tracking data stored on that data source or on another eligible data source. Typically this type of configuration allows an administrator to manage all the tracking and status tables. When using multiple data sources for caching, the cache tracking and cache storage tables stay together on the same data source.

Requirements

- The cache data source and the data source that holds the cache status and tracking tables must be owned by the same user.
- The cache policy requires the use of the same cached data source for status and tracking tables.
- System data sources, such as the file cache or default cache) are not eligible to hold the cache status and tracking tables.
- The data source being used as the cache data source is not eligible to hold the status and tracking tables when attempting to enable this solution.

To enable caching to multiple data sources

1. In Studio, open the data source that you want to use to store the status and tracking tables for your cache. For example, Oracle11.
2. Under Data Source Cache Configuration, select Current data source.
3. For the Status Table and Tracking Table fields, select Browse and browse to existing tables that you want to use or browse and create new tables to hold the data.
4. Save the data source configuration.
5. Open the data source that you want to use to store your cache data. For example, ds_orders.
6. Under Data Source Cache Configuration, select Select a different data source.
7. Browse to locate and specify the data source where you want to store cache data. For example, Oracle11.
8. Verify that the tables indicated in the Status Table and Tracking Table fields are the ones you want to use.
9. Optionally, open the data source and select other status and tracking tables.

   Click Open Data Source. In the Caching section of the Configuration tab, create (execute the DDL), rename or browse to two tables, one for storing cache status data (Status Table) and the other for storing cache tracking data (Tracking Table).
10. Save the data source configuration.
11. Open the table or view resource for which you want to cache data.
12. Select the caching tab.
13. Select Create Cache.
14. Under Storage, select Single Table or Multi-table.

15. For the Data Source field, Browse to the data source which you want to use to hold the cache data. For example, ds_orders.
16. Enable the cache.
17. Save the data source configuration.
18. Refresh the which you want to use to hold the cache data. For example, ds_orders.
19. Enable the cache.
20. Save the data source configuration.
21. To test your configuration, select Refresh Now to populate the cache tables. Open the cache data, cache status and cache tracking tables, validate that each of them contains data and have timestamps that are consistent with the time that you ran the latest cache refresh.

Setting Up Native (Bulk) Caching Options

The TDV native loading option improves cache performance when:

- The cache source and cache target reside on the same TDV data source resource.
- The cache data is being saved to one of the active cache targets listed in the Native Loading Option column of the table in Supported Cache Target Storage Types, page 484.

The native cache loading options are enabled by default. Native loading makes use of functionality that is inherent to the data source to improve the speed of data movement.

If TDV detects that your data source and your cache data target are on the same TDV data source resource, it determines if a direct SELECT and INSERT is the best way to move data or if one of the configurable native loading options might work faster. Most databases have proprietary data movement functionality that is optimized for that database. For example, all Oracle databases come with database link functionality. That database link functionality can be used to provide cache loading performance gains that are not possible when using the TDV JDBC connections to move data.

If both the native database loading and the TDV cache loading fail, the refresh status is set to failed.

Topics in this section	Performance Option
Configuring TDV Native Loading Option for DB2, page 508	Uses the DB2 LOAD utility.
Configuring TDV Native Loading Option for Microsoft SQL Server, page 510	Uses the bcp utility (bcp.exe) for bulk import and export.
Configuring Native Caching for Netezza, page 513	Uses Netezza external tables.
Configuring Native Caching for Oracle, page 515	Uses database links.
Configuring Native Caching Option for Sybase IQ, page 515	LOAD_TABLE statement.
Configuring Native Caching Option for Vertica, page 519	Uses Bulk Load utility.

Configuring TDV Native Loading Option for DB2

For DB2, the TDV native loading option makes use of DB2's LOAD utility. The LOAD utility can quickly load or add data to a table where large amounts of data need to move. LOAD can perform significantly faster than IMPORT because LOAD writes formatted pages directly into the database while IMPORT uses SQL INSERTs. Before attempting to use this method, we recommend that you see if using JDBC batch insert options can give you acceptable performance improvements. If you choose to implement the JDBC functionality, you do not need to configure the DB2 LOAD utility, or change any TDV configuration settings.

The DB2 LOAD utility when working with TDV requires that the DB2 command-line utility be run. How it works with TDV varies by platform:

Platform	DB2 Command-Line Utility Name	Execution Details
Windows	DB2CMD.exe	When it runs, a command window pops up, requires a password, and stays active until the upload completes. While the window is open, the password might be visible.

Platform	DB2 Command-Line Utility Name	Execution Details
UNIX	db2	It can be run as a background process.

Requirements

- The data being loaded must be local to the server.
- Requires advanced DB2 database level configuration

Limitations

This feature is not valid for:

- Binary and BLOB data types
- Kerberos implementations

To configure the DB2 LOAD utility to work with TDV for caching

1. Consult the IBM documentation on configuring and using the DB2 LOAD utility.
2. Install and configure all relevant parts of the DB2 LOAD utility according to IBM's instructions for your platform.

   The client drivers might need to be installed on all the machines that are part of your TDV environment.
3. Verify the full path to the DB2 command-line utility that you want to use to run the DB2 LOAD utility. For example, locate the path to your DB2 SQL command-line client.
4. Open Studio.
5. Select Administration > Configuration.
6. Locate the Enable Bulk Data Loading configuration parameter.
7. Set the value to True.
8. Click Apply.
9. Click OK.
10. Locate the DB2 Command-Line Utility Path configuration parameter.
11. For Value, type the full directory path to the DB2 command-line program. If there are spaces in the pathname, be sure to enclose the path in double quotes. For example:

```
"C:\Program Files\IBM\DB2\Tools\Bin\DB2CMD.exe"
```

12. Click Apply.
13. Click OK
14. Restart your TDV Server.

    Some configuration parameter value changes are applied while the TDV Server is running, but many of them require a server restart to work correctly.
15. Open Studio and locate the DB2 data source for which you want to enable the bulk load feature.
16. Open the data source editor.
17. Select the Advanced tab.
18. Scroll down to locate the Database Alias (Used For DB2 Load) field.

    This property is passed on to the DB2 Load to identify the cataloged database.
19. For Value, type the name of the DB2 database. For example: dvbudb21.
20. Save your changes.
21. Refresh your cache.

Temporary files used to support this feature are created in <TDV_install_dir>\...tmp\cacheloading\db2, all files, data, commands and logs are deleted by TDV after the upload process completes.

Configuring TDV Native Loading Option for Microsoft SQL Server

For Microsoft SQL Server, the TDV native loading option makes use of Microsoft's bulk import and bulk export (bcp) function.

The bulk import and bulk export functionality available caching for Microsoft SQL Server requires the configuration of the Microsoft bcp utility. When the bulk import and bulk export functionality is used instead of the JDBC functionality, performance is improved. The bcp utility (bcp.exe) is a command-line tool that uses the Bulk Copy Program API. The Microsoft bcp utility performs the following tasks:

- Bulk exports data from a SQL Server table into a data file.
- Bulk exports data from a query.
- Bulk imports data from a data file into a SQL Server table.
- Generates format files.

If you choose to implement the JDBC functionality, you do not need to configure the bcp utility, or change any TDV configuration settings.

When the cache source and target exist on the same SQL Server data source, native loading of cache data using a direct SELECT and INSERT can provide significant performance gains. The feature requires that the value of the Enable Bulk Data Loading Studio configuration parameter be set to True.

When configuring data ship or a cache for Microsoft SQL Server, you can configure the bcp utility and then set up access permissions. Both of these processes are described below.

The configuration steps are similar for caching and data ship, for more information see Configuring Data Ship for Microsoft SQL Server, page 611.

Note: In the bcp utility, NULL values are interpreted as EMPTY and EMPTY values are interpreted as NULL. TDV passes the '\0' value (which represents EMPTY) through the bcp utility to insert am EMPTY value into the cache or data ship target. Similarly, TDV passes an EMPTY string ("") to the bcp utility to insert a NULL value into the cache or data ship target.

To configure the Microsoft bcp utility to work with TDV for caching

1. Verify that the bcp.exe utility has been installed and is located in a directory that you can access. Note the full path to the bcp.exe file.
2. Open Studio.
3. Select Administration > Configuration > Data Sources > MS SQLServer Sources.
4. Select the Microsoft BCP utility parameter.
5. For Value, type the full directory path to the bcp.exe. For example:

```
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\bcp.exe
```

6. Optionally to use Windows Authentication, locate and select the DataShip BCP Threshold parameter.

   The default value is 50000. After bcp utility is enabled, if the data shipping cost estimate is 50000 or above, data ship uses the bulk import and export implementation; otherwise JDBC, is used to move the data.
7. Optionally to use Windows Authentication, adjust the value of the DataShip BCP Threshold parameter to values that fit your requirements.
8. Click Apply.
9. Select Administration > Configuration.
10. Navigate to TDV Server > Configuration > Debugging > Enable Bulk Data Loading.

    Or, search for 'bulk' using the Configuration Find field.

11. Set the value to True.
12. Click Apply.
13. Click OK.
14. Stop the TDV Server.
15. From the command line where your TDV Server runs, log in as the domain user.
16. Start the TDV Server.
17. You can now complete the setup of your cache data target.
18. Click OK.
19. Optionally, when running with Windows Authentication, on windows, open Services > TDV Server <version> and select Properties.
20. Select the Login On tab, set the This account fields to a domain user who has access to MSSQL instance.
21. Click OK.

To set up access permissions for SQL Server for caching

1. Consult your SQL Server documentation for how to set the SHOWPLAN permissions for the database.
2. For every SQL Server database that will participate in the caching, grant the SHOWPLAN permissions. For example:

```
GRANT SHOWPLAN
TO <database> [ ,...n ]
```

3. You can now complete the setup in Caching to a File Target, page 496.

To tune the TDV caching behavior when Microsoft SQL Server is the cache target

1. Open Studio.
2. Select Administration > Configuration from the Studio menu bar.
3. Expand the Data Sources > Common to Multiple Source Types >Data Sources Data Transfer.
4. Determine the best value for the Column Delimiter parameter.

   The default is |.
5. Click Apply.

6. Click OK.

Configuring Native Caching for Netezza

For Netezza, the TDV native loading option makes use of the Netezza external tables. You must follow the Netezza documentation for how to set up the external tables on the Netezza data source.

For Netezza you have the following configuration choices:

- Enabling Bulk Load for Netezza, page 513
- Managing NUL Character in Strings for Netezza, page 514

When the cache source and target exist on the same Netezza data source, native loading of cache data using a direct SELECT and INSERT can provide significant performance gains. The feature requires that the value of the Enable Bulk Data Loading Studio configuration parameter be set to True. For Netezza, the parameter value must also be True to enable bulk loading of data cached from procedures.

XML, BINARY, and VARBINARY are not supported by Netezza, any data type related to these data types will not be added to the Netezza cache.

By default, TDV analyzes the table selected for caching to see if DISTRIBUTE can be used to achieve performance gains through parallel processing of the cache query. All keys and indexes specified on the resource are consulted. This is done to determine if the DISTRIBUTE clause was added to the native DDL used to create the cache target tables. For single-table caching, the cachekey column is also added to the DISTRIBUTE clause. If neither keys nor indexes are specified, DISTRIBUTE ON RANDOM is used.

Enabling Bulk Load for Netezza

To enable bulk loading for Netezza

1. Open Studio.
2. Select Administration > Configuration from the Studio menu bar.
3. To tune buffer size for Netezza, expand the Data Sources > Common to Multiple Source Types >Data Sources Data Transfer folder.
4. Determine the best value for the Buffer Flush Threshold configuration parameter.

   This parameter controls how many data rows are written to the buffer file before flushing the buffer. The minimum is 1000 and the default is 10000.
5. Click Apply.

6. Click OK.
7. Select Administration > Configuration.
8. Navigate to TDV Server > Configuration > Debugging > Enable Bulk Data Loading.

   Or, search for 'bulk' using the Configuration Find field.
9. Set the value to True.
10. Click Apply.
11. Click OK.
12. You can now complete the setup of your Netezza cache data target.

Managing NUL Character in Strings for Netezza

Optionally, if you need to move data that has NUL characters, you can use the following procedure to determine how those characters are managed by TDV.

When the Ignore Nul Characters in Strings configuration parameter is true, NUL('\0') characters get discarded by Netezza and rest of the String is loaded. When the configuration parameter is false, NUL characters in Strings are not discarded. The default value is false.

To ignore Nul characters in strings

1. Make sure you have both Modify All Config and Access Tools rights.
2. Log into Studio as the admin user.
3. From the Administration menu, choose Configuration.
4. In the tree pane, navigate to Data Sources > Netezza Sources > Ignore Nul Characters in Strings.
5. Select True.

   When set to false, NUL characters in strings are not discarded.
6. Click Apply.
7. Click OK.

   This Studio configuration change is not immediately propagated to other open instances of Studio connected with this server.
8. Restart the TDV Server.

Configuring Native Caching for Oracle

When you want to use Oracle as your cache target, you can finish setup of the TDV native loading option that uses Oracle's database link functionality to provide better caching performance. Additionally, you can define the TDV parallel cache loading option. This feature is supported for Oracle cache targets.

To set up the native loading option caching data to an Oracle target

1. Make sure that the view that you want to cache:
   - Is a view created from a single TDV data source.
   - Is a view that has a pass-through query that is run (or pushed down) entirely to the data source.
   - The view source can be DB2, Oracle, SQL Server 2008, or Sybase 15.
2. Define database links between your data sources and the Oracle cache target. The database links must be defined using Oracle database tools to directly add the database link using SQL, or by modifying your tnsnames.ora file.
3. Open Studio.
4. Open the Oracle data source that you want to act as your cache target.
5. On the Configuration tab, select the Advanced tab in the Connection Information section.
6. Scroll down and check the Enable Oracle Database Link check box.
7. Type the database link name as you defined it in your Oracle database. The database link path must point to the path of the data that is the source for the cache. Use the Add Database Link button if you want to have more than one database link (to enable caching from multiple sources).
8. Make sure Enable Data Source is checked.
9. Click Add/Remove Resources to open the Data Source Introspection window and find, introspect, remove and view properties of data sources.

   See Retrieving Data Source Metadata, page 189, for details.
10. Click **Test Connection** to make sure the connection works.
11. Save your changes.

Configuring Native Caching Option for Sybase IQ

Depending on the platform you are installing to, your steps might vary. These steps do not detail every step as might be required by Sybase, refer to your Sybase documentation for further details.

To configure Native Cache Loading for Sybase

1. Verify or install a SQL Anywhere Database Client. You can obtain a trial version through the Sybase download site.

   If you don't have SQL Anywhere Database Client and don't want to install one on your Unix system, then you can copy the client into <TDV_install_dir>/sqlanywhere<ver>, because the SQL Anywhere database client has the drivers that are needed for TDV.

2. Locate the following files in the directory where the Sybase client is installed.
   - jodbc.jar
   - libdbjodbc<ver>.so for Unix
   - dbjodbc<ver>.dll for Windows
3. Copy the files to the locations described in the following table.

OS	Copy jodbc.jar Into	Copy dbjodbc<ver>.dll Into
Windows 64	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib\win64
Windows 32	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib
UNIX (32 and 64)	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib

4. Stop your TDV Server.
5. For Unix, set the global LD_LIBRARY_PATH environment variable to point at the SQL Anywhere client libraries folder. TDV uses LD_LIBRARY_PATH to find the Sybase Client library. For example, to temporarily set the variable, type:

```
export
LD_LIBRARY_PATH=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/
```

6. Create an ODBC data source following the guidelines in your Sybase documentation. The following instructions highlight some of the necessary steps depending on your platform.

Platform	Instructions

Unix	1. Create an odbc.ini file. We recommend that you create it under <TDV_install_dir>.
	2. Create a Sybase IQ data source name (DSN) in the odbc.ini file. For work with TDV, the Driver variable is the most important setting, because it points to the SQL Anywhere client that you have installed. For example, data sources named, test1 and test2, would look similar to the following:
	`####################################`
	`SybDB@machine64:cat odbc.ini`
	`[test1]`
	`Driver=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/libdbodbc<v`
	`er>_r.so`
	`host=10.5.3.73`
	`port=2638`
	`uid=dba`
	`PWD=password`
	`DatabaseName=asiqdemo`
	`PreventNotCapable=YES`
	
	`[test2]`
	`Driver=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/libdbodbc<v`
	`er>_r.so`
	`host=10.5.3.74`
	`port=2638`
	`uid=dba`
	`PWD=password`
	`DatabaseName=asiqdemo`
	`PreventNotCapable=YES`
	`####################################`

Platform	Instructions
Windows	1. Start the ODBC Administrator, select Sybase > Data Access > ODBC Data Source Administrator. 2. Click Add on the User DSN tab. 3. Select the Sybase IQ driver and click Finish. 4. The Configuration dialog box appears. 5. Type the Data Source Name in the appropriate text box. Type a Description of the data source in the Description text box if necessary. Do not click OK yet. 6. Click the Login tab. Type the username and password. If the data source is on a remote machine, type a server name and database filename (with the .DB suffix). 7. If the data source is on your local machine, type a start line and database name (without the DB suffix). 8. If the data source is on a remote system, click the Network tab. Click the check box for a protocol and type the options in the text box. 9. Click OK when you have finished defining your data source.

10. On Unix, use the following commands to verify that the DSN is set up successfully:

```
cd sqlanywhere<ver>/bin<ver>
./dbping -m -c "DSN=test1"
```

    A “Ping server successful” message means that the DSN is working and any application can use the DSN to contact the Sybase IQ through the ODBC Driver.

11. On Unix, set the global ODBCINI environment variable to point at the SQL Anywhere odbc.ini file. For example, to temporarily set the variable, type:

```
export ODBCINI=/opt/<TDV_install_dir>/odbc.ini
```

12. Start TDV server.
13. Open Studio.
14. Open your Sybase IQ data source.
15. Select the Advanced tab.
16. Scroll to the end of the Advanced tab.

17. Select and type the following:

Sybase iAnywhere JDBC Driver	Select this check box to enable better performance. This option enables the Sybase IQ specific ODBC LOAD TABLE SQL tool to import data into TDV.
Sql Anywhere Data Source	Enter your ODBC DSN name.

18. Save and close your Sybase IQ data source.

Configuring Native Caching Option for Vertica

By default, TDV is configured to use Vertica's Bulk Load utility or INSERT/SELECT as the native loading mechanism. There are no TDV configuration parameters that you need to set.

Validating that your system is configured to enable native caching

1. Open Studio.
2. Select Administration > Configuration.
3. Search for Enable Native Loading and make sure it is set to True.
4. You can now complete the setup in Caching to a File Target, page 496.

Setting Up the Parallel Cache Option

TDV provides a parallel caching option that is used to maximize performance of data caching. If cache loading performance gains cannot be realized using the TDV native loading option, you can try the TDV parallel option. This option uses statistics derived from a unique numeric key to create multiple partitions that can be used to load the data using parallel processes.

During parallel loading, the reader and writer work in parallel and are not blocked or waiting for one another. The reader reads the table rows into TDV memory and the writer pushes them into target database. You can have multiple readers and corresponding writers that work on different partitions simultaneously. These parallel threads use JDBC-based read write calls.

Requirements

This performance option is available as listed for the Native Cache Target Support column of the table in Supported Cache Target Storage Types, page 484.

It requires some setup beyond TDV configuration parameters. The parallel cache loading option requires that the view that you are caching has:

- One or more user-defined or auto-inferred, simple or composite primary key of numeric or character type.
- For numeric key columns, BOUNDARY cardinality statistics or DETAILED cardinality statistics are needed.
- For character key columns, DETAILED cardinality statistics are needed.

Because partitioning is being determined on the key column, the cardinality statistics help determine how many unique partitions can be loaded in parallel. The number of unique partitions that can be found by the statistics collection determines how many threads can be used to load the data into the cache.

To configure the TDV parallel option for data caching

1. Open Studio.
2. Open a view in Studio that has had caching enabled.
3. Validate that your view has:
   - One or more user-defined or auto-inferred, simple or composite primary key of numeric or character type.

   And that for the following special cases you have the required cardinality statistics:
   - For numeric key columns, BOUNDARY cardinality statistics or DETAILED cardinality statistics are required.
   - For character key columns, DETAILED cardinality statistics are required.
4. Follow the steps in Creating Cardinality Statistics for a View, page 566.
5. Save your work.

The next time your cache is refreshed, this cache loading option is attempted.

Enabling JDBC-Only Cache Loading

If your environment requires that you disable the TDV Native and Parallel Options for cache loading or you need to force the use of JDBC inserts, you need to edit the TDV configuration parameters.

To enable JDBC-only cache loading for Oracle and Netezza data sources

1. Open Studio.
2. Select Administration > Configuration.
3. Search for the Enable Native Loading parameter and set its Value to False.

   If this parameter is disabled, TDV uses JDBC to perform the inserts into the cache table.

   By default this parameter is enabled. For cache targets saved to Oracle, database links are used. For cache targets saved to Netezza, bulk loading options are used to move data.
4. Select the Enable Parallel Loading parameter and set Value to False.
5. Click Apply.
6. Click OK.

Canceling a Cache Refresh that Is Using Native or Parallel Loading

You can cancel a cache refresh process for parallel loading or Netezza.

Note: For Oracle using native loading, the cache refresh process runs to completion; it cannot be stopped after it is started. TDV cleans up the temporary tables automatically.

To cancel a cache refresh process for parallel loading or Netezza

1. Cancel the read request process that has started against the source view.
2. For parallel loading processes, you can cancel the parent request or cancel any of the threads to have the job canceled for all threads.

Defining Cache Refresh Behavior

Cache refresh behavior can be defined to suit your needs. The following topics cover most of the TDV cache refresh behaviors that you can customize:

- Refresh Owner, page 522
- Controlling Cache Refresh Behavior for Individual Resources, page 522
- Controlling Cache Refresh Behavior for Multiple Resources, page 525
- Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530
- Setting Up Pull-Based Incremental Cache, page 531

- Setting Up Push-Based Incremental Caching for Oracle, page 536

Refresh Owner

For Cache Refreshes Started From Studio—TDV always uses the user credentials saved against the data source to perform all runtime cache refresh operations including refresh of the cache data and update of user status in the status table and the internal SYS_CACHES. This is true also even though you might have pass-through logins enabled.

For Cache Refreshes Started From Triggers—If you have triggers defined that you use to start your cache refreshes, the user that initiates the table refreshes is the user that owns the refresh.

The user that is noted for specific actions within the refresh varies depending on your full TDV environment setup, on how many times you have run the refresh, and on whether you look in the Studio Manager, TDV log files, or your database log files. Implementation of TDV is highly customizable. If you are interested in determining which users are involved in the refresh actions at your site, perform testing and track the results.

Controlling Cache Refresh Behavior for Individual Resources

Caches can be refreshed immediately or periodically using Studio. Cache policies can be defined to control the cache refresh behavior for several TDV resources. Caches can also be refreshed programmatically. Keeping data in your cache up-to-date ensures that results are correct. When a view or table resource is refreshed, the entire result set is obtained for the object and written to the cache.

Modifying the query of the view (or table) that you have caching defined on does not automatically update the cache. You must execute a refresh or wait for the next scheduled refresh cycle to get the correct result set.

- Refreshing and Clearing the Cache for One Resource Using Studio, page 522
- Refreshing and Clearing Your Cache Programmatically, page 524

Refreshing and Clearing the Cache for One Resource Using Studio

You can use Studio to refresh your cache data immediately, or you can use Studio to configure regularly scheduled refreshes of your cache data.

For views and tables, the expiration period applies to the whole result set. For procedures, each input variants data has its expiration tracked separately.

Note: If you have multi-table caching defined, TDV first attempts to clear the cache using a TRUNCATE command. If the TRUNCATE command is not supported, a DELETE command is attempted.

To define cache refresh and expiration schedules

1. In Studio, open a view or procedure that has had caching enabled.
2. Select the **Caching** tab.
3. Under Refresh Mode, specify the cache refresh mode. Refreshing a cached resource retrieves data from the sources and clears stale data as specified.

You can refresh the cache immediately by clicking Refresh Now.

Option	Description
Manual	The resource owner or an administrator must manually refresh the cache using Refresh Now or programmatically by invoking the RefreshResourceCache procedure. See the *TDV Application Programming Interfaces Guide*. Any user with the ACCESS TOOLS right who is given appropriate privileges can also refresh the cache manually. If your cache is controlled by a cache policy, using a periodic or programmatic refresh is suggested.
Exactly Once	To refresh the cache just once at a specific time, select, and specify the time to start caching in the set of drop-down boxes in the section labeled **Start on**. The **Hour** field accepts a typed entry of **0**—and when you save the resource, Studio automatically converts the hour to **12** and **AM/PM** to **AM**.
Periodic	To refresh the cache periodically, select **Periodic** and specify in the **Refresh every** section how often to execute the resource for caching: every x number of seconds, minutes, hours, days, weeks, months, or years. In the **Start on** fields, specify the time and date to begin periodic refresh mode.

4. Under Expiration Schedule, select Never expire, or select Expire after and specify the time period. The expiration period applies from the end of a refresh. The Expire after option is disabled if you are using incremental caching.

5. Under Advanced there are a variety of combinations:
   a. If you select Full Refresh Mode, you can define pre-and post-cache actions, as described in Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530.
   b. If you select Incremental Refresh Mode, you can define pre-and post-cache actions, as described in Setting Up Pull-Based Incremental Cache, page 531.
6. Specify when to clear the cache.

Cache Clears Option	Description
when user clears it manually	Clear the cache only when the Clear Now button, an API call, or on cache expiration (Expire after has been selected in the Expiration Schedule section).
when refresh fails	(For full refresh caching only) Selecting this option would clear the cache if a refresh fails. This option allows access to previously cached data during a refresh.
when refresh begins	(For full refresh caching only) Selecting this option would automatically clear the cache before starting a refresh. Any clients attempting to read from the cached data must wait for the new data.
When load fails	(For push-based incremental caching only) Selecting this option would automatically clear the cache if the loading of data did not complete.
When initial load begins	(For push-based incremental caching only) Selecting this option would automatically clear the cache before starting a refresh. Any clients attempting to read from the cached data must wait for the new data.

7. (For procedures only) Define the maximum number of variants you want stored in the cache from the results of the procedure you are caching. Maximum number of unique set of input parameter values. (Default is 32.)
8. Save the cache settings.

Refreshing and Clearing Your Cache Programmatically

If you have chosen to programmatically refresh your cached data, you have the following options:

- TDV provides stored procedures that can be used to clear cache contents, initiate a refresh, and change the state of a cache. You can use a SQL Script to call and perform any caching functions.

- Using the TDV trigger function, it is possible to respond to system- or user-generated events that call a procedure to refresh a cache.
- Administrative Web services can be used to define external programs that clear, refresh, enable, and disable the cache.

The following TDV API calls are available for use from SQL Scripts or Java Procedures, or that can be published for use from clients. Each of the following is accessed in Studio under <localhost>/lib/resource:

- ClearResourceCache(path, type)
- CreateResourceCacheKey(path, type, cacheKey)
- LoadResourceCacheStatus(path, type)
- RefreshResourceCache(path,type)
- UpdateResourceCacheEnabled(path, type, enabled)
- UpdateResourceCacheKeyStatus(path, type, cacheKey, status, startTime, message)

For details, see the *TDV Application Programming Interfaces Guide*.

The following Web services are available for use from clients outside the server. They can be accessed in Studio under Data Services/Web Services/system/admin/resource/operations:

- clearResourceCache (path, type)
- getResourceCacheConfig(path, type)
- refreshResourceCache (path, type)
- updateResourceCacheConfig (path, type, detail, cacheConfig)

Controlling Cache Refresh Behavior for Multiple Resources

Cache policies can be defined to control the cache refresh behavior for several TDV resources. Defining cache policies allows you to reuse cache refresh definitions for multiple TDV resources. The logic inherent to the cache policy behavior manages cache refresh behavior for dependent resources and ensures better data integrity when updated caches for resources that have data dependencies. Keeping data in your cache up-to-date ensures that results are correct. When a view, table, or procedure resource is refreshed, the entire result set is obtained for the object and written to the cache.

Modifying the query of the view (or table) that you have caching defined on does not automatically update the cache. You must execute a refresh or wait for the next scheduled refresh cycle to get the correct result set.

Multi-Table Cache Refresh Limitations

- If you have muti-table caching defined, TDV first attempts to clear the cache using a TRUNCATE command. If the TRUNCATE command is not supported, a DELETE command is attempted.
- If you schedule a reintrospection of the cache data source at exactly the same time as cache refresh, it might lead to errors. The multi-table cache refresh drops and recreates the indexes during cache refresh so it can appear that indexes are missing.

To control cache refresh behavior

1. You must follow the instructions in Defining a Cache Policy, page 526.
2. For each resource that you want to have associated with the policy, follow the instructions in one of the following sections:
   - Assigning Resources to a Cache Policy, page 528
   - Assigning a Cache Policy to a Resource, page 529

Defining a Cache Policy

Cache policies define cache refresh and clearing behavior. The cache policies can then be applied to resources that are being cached.

Cache Policy Limitations

Cache policies are not supported for:

- Procedures with input parameters.
- Pull-based incremental caches when defined on a table or view.
- Push-based incremental caches when defined on a table or view.

To create a cache policy

1. From the Studio resource tree, expand <host>/policy/cache.
2. Right-click and select New Cache Policy.
3. Name the policy.
4. Click OK.
5. Open the new cache policy.
6. Select the Cache Policy Settings tab. For information about the Resources tab, see Assigning Resources to a Cache Policy, page 528.

7. Select the Enable check box to enable this cache policy. If you decide to leave it cleared, you can continue to define the policy, but it will not be active until you select the Enable check box.
8. Accept the default or specify the data source where you want data about the cache policy to be stored. You can use the Browse button to browse through the Studio resource tree of available data sources.
9. Under Refresh Mode, specify the cache refresh mode. Refreshing a cached resource retrieves data from the sources and clears stale data as specified.

   You can refresh the cache immediately by clicking Refresh Now.

Option	Description
Manual	The resource owner or an administrator must manually refresh the cache using Refresh Now or programmatically by invoking the RefreshResourceCache procedure. Any user with the ACCESS TOOLS right who is given appropriate privileges can also refresh the cache manually. If your cache is controlled by a cache policy, using a periodic or programmatic refresh is suggested.
Exactly Once	To refresh the cache just once at a specific time, select, and specify the time to start caching in the set of drop-down boxes in the section labeled Start on. The Hour field accepts a typed entry of 0—and when you save the resource, Studio automatically converts the hour to 12 and AM/PM to AM.
Periodic	To refresh the cache periodically, select Periodic and specify in the Refresh every section how often to execute the resource for caching: every x number of seconds, minutes, hours, days, weeks, months, or years. In the Start on fields, specify the time and date to begin periodic refresh mode.

10. Under Expiration Schedule, select Never expire, or select Expire after and specify the time period. The expiration period applies from the end of a refresh. The Expire after option is disabled if you are using incremental caching.
11. Optionally, you can define pre-and post-refresh actions. As described in Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530.

12. Specify when to clear the cache:

Cache Clears Option	Description
when user clears it manually	Clear the cache only when the Clear Now button, an API call, or on cache expiration (Expire after has been selected in the Expiration Schedule section).
when refresh fails	(For full refresh caching only) Selecting this option would clear the cache if a refresh fails. This option allows access to previously cached data during a refresh.
when refresh begins	(For full refresh caching only) Selecting this option would automatically clear the cache before starting a refresh. Any clients attempting to read from the cached data must wait for the new data.

13. Save the cache settings.

Assigning Resources to a Cache Policy

You can assign one or more resources to a cache policy to have their cache refreshes managed by the same refresh definition. After a resource is assigned to a cache policy and the cache policy is saved, TDV sets the definitions on Caching tab of that resource to those defined by the cache policy. If caching was not enabled for the resource that you assign to the cache policy, it is enabled and the majority of the fields on the Caching tab are dimmed to indicate that they are under the control of the cache policy.

To associate a resource with a cache policy

1. Make sure that cache policies have been defined following the instructions in Defining a Cache Policy, page 526.
2. In Studio, open a cache policy.
3. Select the Resources tab.
4. Add resources to the list to be managed by this cache refresh policy in one of the following ways:
   - Select one or more resources from the Studio resource tree and drag them into the Resources text field.
   - Click the green plus button and use the Select Resources dialog to navigate to the resource you want to add.
5. Save your cache policy.

Studio does validation of the policy definitions for the resources in the list and issues messages if there are things you need to fix.

6. Fix resource definitions if prompted by any Studio messages.
7. Save your changes.

Assigning a Cache Policy to a Resource

Defining cache policies allows you to reuse cache refresh definitions for multiple TDV resources. The logic inherent to the cache policy behavior manages cache refresh behavior for dependent resources and ensures better data integrity when updated caches for resources that have data dependencies. If you did not add the resource to the cache policy on the **Resources** tab, you can follow these steps to associate a specific resource to a cache policy.

Note: If you have multi-table caching defined, TDV first attempts to clear the cache using a TRUNCATE command. If the TRUNCATE command is not supported, a DELETE command is attempted.

Cache Policy Limitations

Cache policies are not supported for:

- Procedures with input parameters.
- Pull-based incremental caches when defined on a table or view.
- Push-based incremental caches when defined on a table or view.

To associate a resource with a cache policy

1. Make sure that cache policies have been defined following the instructions in Defining a Cache Policy, page 526.
2. In Studio, open a view, table, or procedure that has had caching enabled.
3. Select the Caching tab.
4. Locate the Cache Policy field.
5. Select the cache policy that you want to be used to control cache refresh and clearing behavior for this resource. You can use the Open Policy button to open the policy and review or change its settings as necessary.
6. (For procedures only) Define the maximum number of variants you want stored in the cache from the results of the procedure you are caching. Maximum number of unique set of input parameter values. (Default is 32.)
7. Save your resource.

Note: The **Refresh Now** button appears enabled for each resource that is not associated with a cache policy. If the resource is associated with a cache policy, however, the button will appear disabled.

Defining Pre- and Post-Actions for a Full Refresh Mode Cache

When caching data, you might want to have actions take place before or after the cache data is refreshed. These pre- and post-actions are available for the automatic, single table, or multi-table caching modes.

For example, before the cache is refreshed you might want to send email notifications that the data is about to be refreshed. After the cache is refreshed, you might want to define or update the indexes associated with the data.

Within Studio, you can define procedures or scripts that have no input or output parameters to act as your pre- and post-actions. They can, however, call other procedures to execute subtasks within the pre-refresh or post-refresh operation.

The pre and post procedures can be a SQL script procedure or a custom Java procedure. They cannot be an XSLT transform or XQuery procedure.

To define a pre- or post-refresh actions

1. Create a new SQL script or Java procedure using the instructions in Java Procedures, page 277 or Adding a Custom Java Procedure, page 167.

   To use an existing SQL script or procedure locate the script or procedure within the Studio resource tree.

2. Make sure that the script or procedure does not have input or output parameters.

3. Evaluate the following TDV API calls for use in Studio under <localhost>/lib/util:

   — GetEnvironment—Can be used to return one or more of the system properties to your Pull-Based incremental caching SQL script

   — GetProperty—Can be used to return one or more of the system properties to your Pull-Based incremental caching custom Java procedure.

4. Review other functions under the lib directory to determine if they can help you achieve the action that you want for your pre- or post-action. Other functions of possible interest might include:
   - CreateResourceCacheKey is used to create a cache key for a given resource.
   - TestDataSourceConnection is used to test to see if a data source's connection is operational.
   - ClearResourceCache(path, type) is used to clear the cache on a resource.
   - UpdateResourceCacheKeyStatus(path, type, cacheKey, status, startTime, message) is used to update the cache key for the specified resource.
   - GetResourceCacheStatusProcedure is used with CreateResourceCacheKey and UpdateResourceCacheKeyStatus to support external cache loading. Returns Cache status information for a given cache key.
5. Save the script or procedure.
6. In Studio, open a view or procedure that has had caching enabled for which you want to assign pre- and or post-actions.
7. Select the Caching tab.
8. Under Advanced, choose Full Refresh Mode.
9. For the Pre-Refresh or Post-Refresh Action fields, or both, specify a procedure that exists in the Studio resource tree and has no parameters.
10. Save the cache settings. After cache settings are saved, each cached resource appears with a lightning-bolt icon in the Studio resource tree to show that the resource is cached.
11. If you have data type incompatibilities between your view and your data storage type, see "Cache Data Type Mapping" in the *TDV Reference Guide*.

Setting Up Pull-Based Incremental Cache

Pull-based incremental caching can be set up using the TDV caching mechanism with two scripts. With pull-based incremental caching the user or client application must make a request for their copy of the cache to be synchronized with the centralized cache copy. Because you define the scripts used to create and refresh the cache, you can make the scripts as complicated or as simple as your situation requires. Your scripts can be defined using SQL or using custom Java procedures. The procedure must have logic to identify what rows have changed.

This topic covers the generalized steps for creating the scripts necessary for the pull-bases method of defining incremental caching and provides the following sample scripts:

- Pull-Based Incremental Cache Initialization Sample Script, page 534
- Pull-Based Incremental Cache Refreshing Sample Script, page 535

To set up pull-based incremental caching

1. Locate or create a reliable and unique identifier within your data source that you can use to determine how data has changed since the last cache refresh. This unique identifier could be a system change number, a log sequence number (LSN), or a TIMESTAMP. Typically the cachekey column can be used to determine what data has been updated.
2. Create a new SQL script or Java procedure using the instructions in Java Procedures, page 277 or Adding a Custom Java Procedure, page 167.

   To use an existing SQL script or procedure locate the script or procedure within the Studio resource tree.
3. Make sure that the scripts or procedures for the cache have one VARCHAR output parameter.
4. Evaluate the following TDV API calls for use are in Studio under <localhost>/lib/util:

   — GetEnvironment—Can be used to return one or more of the system properties to your Pull-Based incremental caching SQL script

   — GetProperty—Can be used to return one or more of the system properties to your Pull-Based incremental caching custom Java procedure. Specifically, you can use it to acquire the server ID.

   The system caching properties that can be interacted with include:

   — System.CACHED_RESOURCE_PATH

   — System.CACHED_RESOURCE_TYPE

   — System.CACHED_RESOURCE_PARAM_KEY

   — System.CACHE_DATASOURCE_PATH.

   — System.CACHED_RESOURCE_CACHE_KEY

   — System.CACHED_RESOURCE_BUCKET_PATH.

   — System.CACHED_RESOURCE_REFRESH_OUTCOME

   — System.CACHED_RESOURCE_ERROR_MESSAGE

   — System.CACHED_RESOURCE_INCREMENTAL_MAINTENANCE_LEVEL

5. Review other functions under the lib directory to determine if they can help you achieve the action that you want for your pull-based incremental caching. Other functions of possible interest might include:
   - CreateResourceCacheKey() is called to generate a new cachekey value.
   - LoadResourceCacheStatus() is called to inform the server of the in-progress refresh.
   - LoadResourceCacheStatus() is called to inform the server of the newly active data.
   - TestDataSourceConnection is used to test to see if a data source's connection is operational.
   - ClearResourceCache(path, type) is used to clear the cache on a resource.
   - UpdateResourceCacheKeyStatus(path, type, cacheKey, status, startTime, message) is used to update the cache key for the specified resource.
   - GetResourceCacheStatusProcedure is used with CreateResourceCacheKey and UpdateResourceCacheKeyStatus to support external cache loading. Returns Cache status information for a given cache key.
6. Consider using some of the following logic to control caching logic:
   - The status table is queried to find the currently active cachekey. The cachekey is in the row with the server ID, resource path, and status column with value 'A'. The INSERT, UPDATE, and DELETE operations can be performed on the data table using the cachekey as a filter to avoid updating any other keys.
   - Perform INSERTs and other operations to create new rows in the storage table as appropriate, making sure all such rows have the new cachekey value. This should be performed on an independent transaction so it can be committed when done.
   - Perform INSERTs to the status table with the server ID, resource path, cachekey, the status set to 'I', and the starttime set to the current time to indicate an in-progress refresh. This should be performed on an independent transaction so it can be committed immediately.
   - UPDATE all rows in the status table with the server ID and resource path that have status 'A' to have status 'C'. This marks the previous active cache data for clearing. Then UPDATE the row into the status table with the serverID, resource path, and cachekey to set the status set to 'A' and finishtime to the current time. This will indicate that this cache key is the new active one. This should be performed on an independent transaction so it can be committed immediately.
7. Save the script or procedure.

8. In Studio, open a view or procedure that has had caching enabled for which you want to define Pull-Based incremental caching.

   Note: If your cache objects were created prior to the current version of TDV, you might need to recreate them.

9. Select the **Caching** tab.
10. Under **Advanced**, choose **Incremental Refresh Mode** to enable incremental caching that can be refreshed on demand.

    Incremental refresh caching only updates the client version of the caches when the client requests the update.

11. Specify values for the following fields:

    — Initialize the cache using—Specify a procedure or script that exists in the Studio resource tree and has one output parameter. This script will be used to create the initial cache.

    — Refresh the cache using—Specify a procedure or script that exists in the Studio resource tree and has one output parameter. The output parameter should be a of VARCHAR data type.

12. Save the cache settings. After cache settings are saved, each cached resource appears with a lightning-bolt icon in the Studio resource tree to show that the resource is cached.
13. If you have data type incompatibilities between your view and your data storage type, see "Cache Data Type Mapping" in the *TDV Reference Guide*.

Pull-Based Incremental Cache Initialization Sample Script

The following sample script is used to create the cache table. It has one IncrementalMaintenanceLevel output parameter of type VARCHAR.

```
PROCEDURE InitalCache(OUT IncrementalMaintenanceLevel VARCHAR)
BEGIN
DECLARE cacheKey BIGINT;
DECLARE maxI BIGINT;

/* 1. Retrieve cache key from request environment */
CALL /lib/util/GetEnvironment('System.CACHED_RESOURCE_CACHE_KEY',
cacheKey);
CALL /lib/debug/Log('cachedResourceCacheKey = ' || cacheKey);

/* 2. Determine initial snapshot level */
SELECT {option no_data_cache} MAX(i) INTO maxI FROM
/shared/INCREMENTAL_CACHING/INCR_CACHE_TEST;
SET maxI = COALESCE(maxI, 0);

/* 3. Load cache target table */
INSERT INTO
/shared/INCREMENTAL_CACHING/incr_cache_test_target
```

```
SELECT {option disable_data_cache}
cacheKey, S.*
FROM
/shared/INCREMENTAL_CACHING/INCR_CACHE_TEST S
WHERE
i <= maxI;

/* 4. Return incremental maintenance level */
SET IncrementalMaintenanceLevel = CAST(maxI AS VARCHAR);
END
```

Pull-Based Incremental Cache Refreshing Sample Script

The following sample script is used to request that a local copy of the cache be refreshed with the latest data from the cache table. It has one IncrementalMaintenanceLevel output parameter of type VARCHAR.

```
PROCEDURE DeltaLoader(OUT IncrementalMaintenanceLevel VARCHAR)
BEGIN
DECLARE cacheKey BIGINT;
DECLARE maxI BIGINT;

/* 1. Retrieve cache key from request environment */
CALL /lib/util/GetEnvironment('System.CACHED_RESOURCE_CACHE_KEY',
cacheKey);
CALL /lib/debug/Log('cachedResourceCacheKey = ' || cacheKey);

/* 2. Retrieve incremental maintenance level from request
environment */
CALL
/lib/util/GetEnvironment('System.CACHED_RESOURCE_INCREMENTAL_MAINT
ENANCE_LEVEL', IncrementalMaintenanceLevel);
CALL
/lib/debug/Log('cachedResourceIncrementalCacheMaintenanceLevel  =
' || IncrementalMaintenanceLevel);

/* 3. Determine next level */
SELECT {option no_data_cache} MAX(i) INTO maxI FROM
/shared/INCREMENTAL_CACHING/QA/"db-lab-9"/QAN/INCR_CACHE_TEST;
SET maxI = COALESCE(maxI, 0);

/* 4. Refresh cache target table */
INSERT INTO

/shared/INCREMENTAL_CACHING/QA/"db-lab-9"/QAN/incr_cache_test_targ
et
SELECT {option disable_data_cache}
cacheKey, S.*
FROM
/shared/INCREMENTAL_CACHING/QA/"db-lab-9"/QAN/INCR_CACHE_TEST S
WHERE
i > CAST(IncrementalMaintenanceLevel AS BIGINT) AND i <= maxI;

/* 5. Update incremental maintenance level */
SET IncrementalMaintenanceLevel = CAST(maxI AS VARCHAR);
```

```
CALL /lib/debug/Log('IncrementalMaintenanceLevel for successful
run of  DeltaLoader script= ' || IncrementalMaintenanceLevel);

EXCEPTION
      ELSE
            --Log the exception
            CALL /lib/debug/log('Exception raised in the delta
loader script');
     CALL /lib/debug/log('Exception is : ' ||
CURRENT_EXCEPTION.NAME || ': ' ||
CURRENT_EXCEPTION.MESSAGE);

    --Don't advance the incremental maintenance level on a failure
      CALL
/lib/util/GetEnvironment('System.CACHED_RESOURCE_INCREMENTAL_MAINT
ENANCE_LEVEL', IncrementalMaintenanceLevel);
    CALL /lib/debug/Log('IncrementalMaintenanceLevel after
exception in DeltaLoader script= ' || IncrementalMaintenanceLevel);

END
```

Setting Up Push-Based Incremental Caching for Oracle

Push-based incremental caching can help to achieve performance gains when using Oracle data sources. Data consumers can pull the latest data from incrementally maintained and synchronized caches that reduce impact on your transaction servers. Because this type of caching requires very specific set-up and configuration, it is described in Push-Based Incremental Caching, page 629.

Cache Maintenance

Over time your caching needs might change. This section details some of the common cache maintenance topics:

Indexing Your Cache

TDV provides several options for you to index your cached data. Indexing your data can improve performance when certain columns are commonly used for filtering and viewing a subset of the data from the cache. Adding an index can slow the performance of cache refreshes, but can improve the performance of cache result retrieval.

Depending on what cache options you have chosen to implement, different indexing options are available to you:

- For a full refresh type of cache, you can specify scripts to run before and after a cache refresh. You could include logic in each of those scripts to drop and/or create indexes on your cached data.
- For a full refresh cache that uses the multi-table option, you can configure TDV to index your cached data automatically with each refresh. This selection requires that you provide indexing definitions on the Index tab for the table or view that you are caching.
- For pull-based incremental caching, you can add logic to the incremental caching scripts to define additional indexing behavior.
- For TDV caching options that don't directly support indexing, you can create indexes directly on your cached data using the database tools available for your specific cache target. This solution might also require that you consider what processes might be necessary to ensure that the indexes are still valid after a cache refresh occurs.

Indexing-specific columns in the result set of a cached view yields performance benefits. TDV also provides the ability for you to define a script or procedure that can be used to initiate or update and index on your cached data.

You can filter data going to the cache table using the cachekey column. Because the cachekey column is a unique value for a given cache snapshot, when you write custom scripts to index the cache data, we recommend that you make the cachekey the first column of the index. This is especially true if you have many procedure cache variants and commonly used queries do not return a large subset of rows.

To index your cache if full-refresh multi-table caching is enabled

1. Refer to the instructions in:
   — Creating a Multiple Table Cache on a Database Target, page 501
2. Make sure that the Indexes tab has the index defined as you need.

For information using the Indexes tab, see Use Indexes, and Primary and Foreign Keys, to Improve Query Performance, page 573.

To index your full refresh cache

1. Refer to the instructions in the following topic:
   — Defining Pre- and Post-Actions for a Full Refresh Mode Cache, page 530
2. Add logic to manage and define the indexes.
3. Test your performance to ensure that performance goals are reached for cache indexes.

To index your pull-based incremental cache

1. Refer to the instructions in the following topic:
   — Setting Up Pull-Based Incremental Cache, page 531
2. Add logic to manage and define the indexes.
3. Test your performance to ensure that performance goals are reached for cache indexes.

To index your cache data directly on the cache target

1. Refer to the instructions in the following topic:
   — Setting Up Caching, page 494
2. Define the indexes using the database tools available for your cache target database type.
3. Determine what other scripts or process changes might be necessary to make sure that the index is still functional after a cache refresh.
4. Test your performance to ensure that performance goals are reached for cache indexes.

Managing Configuration Changes and Cache Behavior

Most changes to the table, view, or procedure that you are caching, or changes to your cache data storage definitions, require that you reevaluate your cache configuration. Certain changes could result in missing cache data, which would result in temporary errors for any application attempting to access the data in the cache. Performing an explicit clear or refresh action following a configuration change is recommended.

The following configuration changes are managed automatically by Studio:

- If you rename an object that you are caching or any of its parent folders, your cached tables are updated.

- If you are using the automatic caching option, Studio drops the old files and creates any necessary new files automatically.

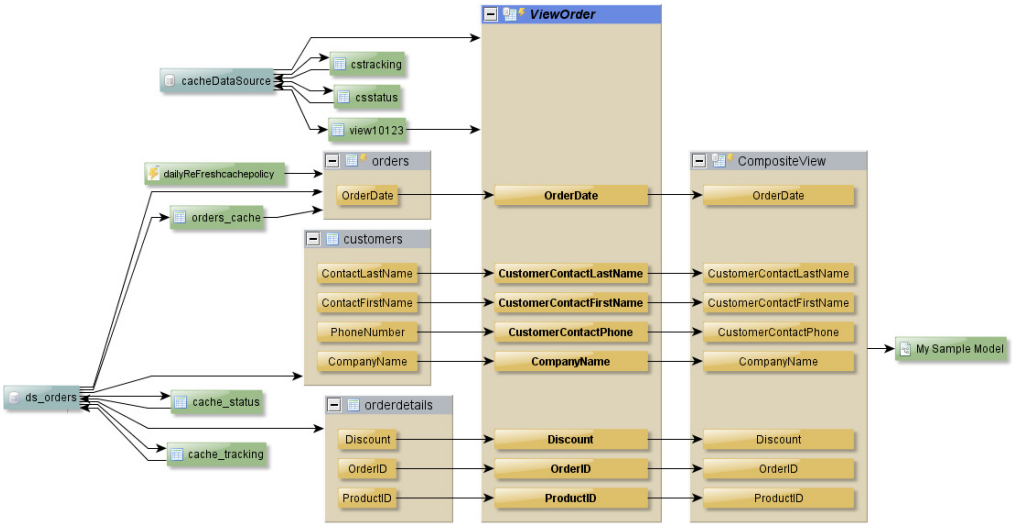
If you change any of the following, be sure to fully reevaluate your caching configuration:

- The storage data source type, physical location, name, number of columns, or the column data types.
- The cache status table physical location, name, or other metadata.
- The number of columns or a column data type definition for the view you are caching.
- The name or number of parameters for a procedure you are caching.

Displaying the Dependencies of a Cached View

When you are working on a cached view, it is useful to know what resources are involved so you can understand the behavior of the view and the cache. The Lineage panel in Studio shows a resource's dependencies, references, and any associated cache policies. The Lineage panel is also available for cached views.

The Lineage panel displays the view's dependencies on its left and the resources that depend on it on its right. Cache tables are shown in green.



To display the dependencies and references of a cached view

1. Use one of these methods to open the Lineage panel:

Method	Description
Right-click a view that has caching enabled in the resource tree and select Open Lineage.	The Lineage panel opens in the workspace to the right. It displays all the resources related to the view in a graphical format. Use this method if the view has lots of objects because the entire workspace is used to display the dependencies and references. This lineage display also allows you to easily navigate the related resources and saves the history of the navigation.
Open a view that has caching enabled and click the Open Lineage Panel toolbar button.	The Lineage panel opens in the lower pane of the view's editor and displays all the resources involved with this view in a graphical format.

2. For more information on how to interact with the lineage graph editor, see Working with the Data Lineage Graph, page 450.

Displaying the Dependencies of a Cache Policy

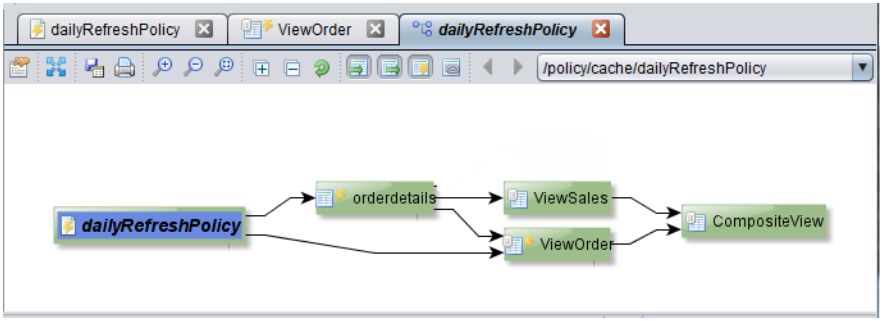
When you are working on a cache policy, it is useful to know what resources are involved so you can understand the behavior of the cache refresh action. The Lineage panel in Studio shows a resource's dependencies, references, and other data points for the associated cache policy.

To display the dependencies and references of a cache policy

1. Open your cache policy.
2. Select the Cache Policy Settings tab.
3. Scroll to the Lineage section.
4. Click Open Lineage.

   The Lineage panel opens in the workspace to the right. It displays all the resources related to the view in a graphical format. Use this method if the view has lots of objects because the entire workspace is used to display the dependencies and references. This lineage display also allows you to easily navigate the related resources and saves the history of the navigation.

Cache tables and policies are shown in green.



5. For more information on how to interact with the lineage graph editor, see Working with the Data Lineage Graph, page 450.

Managing Import and Export Cache Information

Cache information can be exported and imported to Studio using the export and import windows in Studio or the backup_export and backup_import command-line syntax.

- Exporting Your Cache Information, page 541
- Importing Your Cache Information, page 541

Exporting Your Cache Information

When exporting your cache information using either the UI or the backup_export method is fairly straightforward.

To export cache information

1. Identify all of the resources included in the caching environment that you want to export to a CAR file. For example, you might need to export a data source, a view, and one or more cache policies.
2. Export the various items using the Studio Export UI or the backup_export syntax.

Importing Your Cache Information

When importing your cache information using either the UI or the backup_import method is fairly straightforward.

To import cache information

1. Identify the CAR file that you want to import.
2. Import the CAR using the Studio Import UI or the backup_import syntax.

   If your CAR file includes cache policies, import the CAR from the Studio Desktop directory level. Cache policies must be save and used from the /policy/cache Studio directory.
3. If necessary, rebind cache resources after importing the CAR.

Caching Tips from an Expert

Destroying a File or Default Cache

Using the Studio Caching tab, there is a way to destroy the file cache or default PostgreSQL database cache. For other cache data sources in use, if the database server is forced to terminate due to a systems problem, then orphaned files could be left on disk.

For single and multi-table database caching, the actions described here will not drop the physical database tables.

To destroy a cache

1. Locate and open the resource for which you want to destroy the associated cache.
2. Select the Cache tab.
3. Clear the Enable check box and Save to disable the cache.
4. On the upper right corner of Caching tab, locate the Destroy Cache icon.
5. Click it and click Yes.

The file or tables specific to the resource that you selected are destroyed. However, the metadata associated with the files might continue to display in Studio.

Managing Cache Status Table Probe Row Conflicts

On very rare occasions when there is a connection issue to the database where the cache_status table is, the status of a cache refresh can remain in the Probe state. The cache process should only be in the Probe state for a very short period of time, but if there is a disconnect while in that state, the cache can be orphaned in that state until the TDV server is restarted.

TDV has a configuration parameter that can be used to eliminate this situation.

To manage caching conflicts

1. Open the Studio Administration menu Configuration option.
2. Modify the value of the Wait Time For Probe Conflicts configuration parameter.

   Adjust the value in seconds to wait for probe row conflicts if necessary. Minimum value is 10 seconds.
3. Save your changes.

Managing Unresponsive Cache Tables

The startup of the TDV Server can be significantly slow if there is a problem when accessing your cache tables. This is caused because the TDV Server process reads all the cache status tables as part of the startup routine.

TDV has a configuration parameter that can be used to eliminate this situation. When it is set to a non-zero value the server attempts to connect to each cache database within the specified timeout (in milliseconds) before performing any further cache status or cache data maintenance.

To manage unresponsive cache tables for TDV Server restart

1. Open the Studio Administration menu Configuration option.
2. Modify the milliseconds value of the Scan Cache Database Connection Timeout on Server Startup configuration parameter.

   If set to a non-zero value the server attempts to connect to each cache database within the specified timeout (in milliseconds) before performing any further cache status or cache data maintenance.

Set to a value high enough to work well for your environment and network characteristics. Setting it too low renders caches unusable by TDV.

3. Save your changes.

The setting will take effect on the next server restart.

Managing Open Connection Threads

The driver properties are used to specify connection timeout settings as required by the specific driver. By specifying the properties that are used by your specific data source, you can avoid situations where connections are left open indefinitely. This will prevent TDV threads from locking up on requests that need to establish a database connection.

Because each database has different connection properties, you will need to become familiar with them to determine which property to set and what values are appropriate for your environment.

To manage the connection threads

1. Open the data source editor for the data source that has been identified as having connection threads that stay open too long.
2. Open the Advanced tab.
3. Edit the Connection Properties—JDBC Connection Properties to define connection timeout intervals that are appropriate for your environment.

Managing Buckets

Occasionally when performing a cache refresh, you might encounter a situation where you receive a message stating:

```
All buckets for the cached resource are in use
```

Typically this message is displayed because there are several open client connections that are accessing the data in the cache buckets.

To manage the buckets

1. Open Studio, and open the Studio Manager.
2. Open Sessions.
3. Determine if any client connections are accessing the view which is cached.

4. Determine which of the following options is best for your situation:
   - Wait for the client connections to finish before rerunning the cache refresh.
   - Increase the number of buckets to accommodate concurrency.
5. To modify the Number of Caching Tables defined for your cache, see Enabling Caching to Multiple Tables, page 502.
6. Save any changes you have made.
7. Rerun the cache refresh.

TDV Configuration Parameters

Studio provides configuration parameters that allow modification of values and behavior. The Studio Configuration window provides access to configuration parameters. Default values configure TDV for a development environment where a moderate-size team prepares prototypes for test and later creates a production deployment.

Viewing or Editing Configuration Parameters

The Configuration window lets you view all of the configuration parameters for TDV.

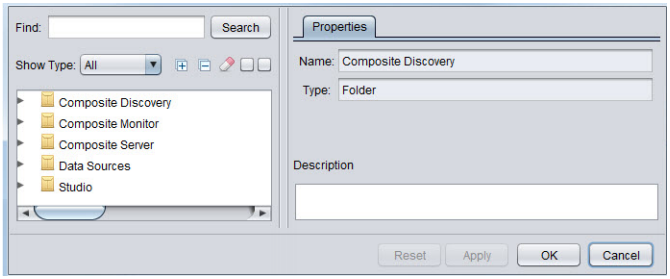
Special characteristics of parameters are provided in their names or descriptions:

- Configuration parameters with (Current) in their names are read-only and exist only to display current settings.
- Configuration parameters with (On Server Restart) in their names change to your new values only when TDV is restarted.
- Some configuration parameters have descriptions that contain "This value is locally defined. It will not be altered when restoring a backup and will not be replicated in a cluster." See the *Active Cluster Guide* for more information about replication in a cluster.

To view or edit configuration parameters

1. For Viewing, make sure you have Access Tools and Read All Config rights. For editing, make sure you have Access Tools and Modify All Config rights.
2. Select Administration > Configuration from the Studio Modeler toolbar to open the Configuration window.

The left panel shows a list of folders indicated by a folder icon. You can expand the folders to view the resource tree of configuration parameters. The right panel displays properties for the currently selected parameter.



3. Click the **Expand All** button to display the full hierarchy of folders and parameters, or click the triangle to the left of a folder name to expand only that folder.

   You can click the **Collapse All** button to return the display to the five top-level folders.

4. Select a parameter in the left pane to view its properties in the right pane.

   If the parameter value cannot be changed, its icon is dimmed.

Type	Possible Actions
Choice	Click the radio button indicating the opposite choice.
List	Click add button in the Value area to add a value to the list. Click delete button adjacent to a value to remove it.
Map	Click plus in the Value area to add a key-value pair: one field for a Key, the other field for a Key Value. Click delete button adjacent to a key-value pair to remove it.
Number	Type a new number.
Optional password	Type a password in the Password field and the identical password in the Confirm field. Click Apply or OK to make the password required. The Confirm field returns to blank for the user to type a matching password.
Required password	Type the required password in the Confirm field. The Password Value is a string of asterisks (*) representing the password that has been set.
Text	Type a new string of characters.

5. If you want to show only the parameters that have pending changes, check the second check box from the right under the Search button.

   The check box tooltip says Show only modified files.

   An asterisk (*) is displayed between a parameter icon and its name if a new value is pending.

6. To apply none, some, or all of the pending changes, do one of the following.

To...	Do This...
Clear the pending change to the highlighted parameter.	Click Reset.
Clear all pending changes.	Click clear.
Apply all pending changes without closing the Configuration window.	Click Apply. The changes are no longer considered pending.
Apply all pending changes and close the Configuration window.	Click OK.
Close the Configuration window without applying any pending changes.	Click Cancel.

Finding Parameters in the Configuration Window

The Configuration window contains a Find field and Search button to help you find parameters.

The Configuration window also contains a parameter-type filter and a toolbar.

Note: See also Viewing or Editing Configuration Parameters, page 547.

To find a configuration parameter and its values

1. Select Administration > Configuration from the Studio Modeler toolbar to open the Configuration window.
2. Use one of the following methods to search for the parameter.

To...	Do This...
Display only parameters of a specific type.	Select that type from the Show Type drop-down list. The Show Type drop-down list is explained earlier in this topic.
Return to displaying all parameters.	Select All from the Show Type drop-down list.
Search by parameter *name.*	Type part or all of the name in the Find field and click Search.
Search by parameter *type*.	Select the parameter type from the Show Type drop-down list, or select All to show all types.
Search by parameter *value* or *description.*	Check the Search through descriptions instead of names check box.
Display only parameters that have pending changes	Check the Search Only Modified Parameters check box.
Display the full hierarchy of folders and parameters.	Click the Expand All button.
Display only the top-level configuration parameter folders.	Click the Collapse All button.
Display the folders and parameters immediately below a folder.	Click the triangle to the left of the folder, or double-click the folder name.

3. Highlight the parameter to view its properties in the right pane.
4. Click OK or Cancel to close the window.

Performance Tuning

This topic describes ways to fine-tune query execution to enhance performance.

About Performance Tuning in TDV

You can tune TDV performance in several ways. Some performance tuning topics are covered in this topic, some elsewhere. The following table summarizes and links to the sections that describe performance tuning.

Tuning Tool	Summary	Topic Reference
Data caching	Saving local snapshots of data can improve the performance of some design tasks. Caching temporarily stores a copy of frequently accessed source data.	• TDV Caching, page 465

Tuning Tool	Summary	Topic Reference
Generating query execution plans	The execution plan shows how TDV Server plans to execute the SQL for a view. It can help you understand what you might do to enhance query performance.	• Working with the SQL Execution Plan, page 553 • Generating and Displaying a SQL Execution Plan, page 554 • Execution Plan Nodes in the Tree Panel, page 555 • Updating the Query Execution Plan, page 562 • Viewing How Much Data was Processed by a Query, page 563 • Refreshing All Execution Plan Caches, page 565
Creating cardinality statistics	Cardinality statistics show the distribution of values in a table or view, and estimate the number of rows in the query result. This can help you improve the query plan.	• Creating Cardinality Statistics for Cost-Based Optimization, page 565 • Using TDV API Procedures to Refresh or Cancel Resource Statistics, page 572
Defining indexes, and primary and foreign keys	An index, or a primary or foreign key, can make data easier to find.	• Use Indexes, and Primary and Foreign Keys, to Improve Query Performance, page 573
Using joins	Join strategies can significantly affect query performance.	• Types of Joins, page 573 • SQL Join Reordering, page 576 • Semijoin Optimization Option, page 579 • Star Schema Semijoin, page 587
Using SQL hints	You can use these when the default behavior of the query optimizer does not result in the best query performance.	• Using Oracle SQL Optimizer Hints, page 588

Tuning Tool	Summary	Topic Reference
Using nested aggregates	If a data source does not support nested aggregates, you can improve performance by letting TDV know not to look for them.	• Tuning Nested Aggregate Function Behavior, page 589
Using data ship	Data ship optimization analyzes queries to determine if performance can be improved by temporarily transferring a small amount of data to a data source target.	• Data Ship Performance Optimization, page 597

Working with the SQL Execution Plan

TDV provides ways for you to generate and evaluate the execution plans for SQL queries. The execution plan displays characteristics of the overall request, and of each request node, depending on which you highlight. The request hierarchy is displayed on the left (the Tree Panel), and the plan information is displayed on the right (the Details Panel).

When you execute a particular SQL query for the first time, the TDV Server caches the query execution plan so it can reuse it if you resubmit the query later.

Working with the SQL execution plan is described in these sections:

- Generating and Displaying a SQL Execution Plan, page 554
- Execution Plan Nodes in the Tree Panel, page 555
- Execution Plan Query Attributes in the Details Panel, page 558
- Execution Plan Node Attributes in the Details Panel, page 560
- Updating the Query Execution Plan, page 562
- Viewing How Much Data was Processed by a Query, page 563
- Refreshing All Execution Plan Caches, page 565

You can also explicitly gather and cache the statistics for a query, such as number of rows in each table. The execution plan helps the TDV Server make better decisions about how to process the query. However, if the statistics change, you should regenerate the query plan so TDV uses up-to-date information. See Creating Cardinality Statistics for Cost-Based Optimization, page 565.

Generating and Displaying a SQL Execution Plan

The SQL execution plan shows how TDV plans to execute a SQL query, both on the overall (request) level and on the level of each query node. Execution plan information can help you enhance query performance.

Note: Execution plan and query plan refer to the same thing.

You can display an execution plan in one of two ways:

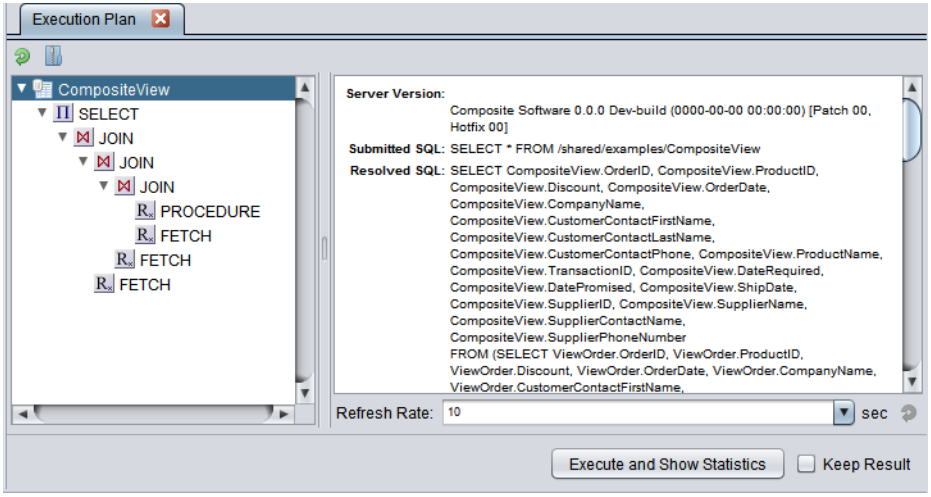
- For the current query, you can show the execution plan in Studio Modeler.
- For any past query, you can show the query plan in Studio Manager.

By default, Studio opens in the Modeler, the top vertical tab on the left side of the Studio window. To open Studio Manager, click the middle vertical tab.

To display the SQL execution plan in Studio Modeler

1. Open a query in Studio Modeler.
2. Click Show Execution Plan on the editor toolbar.

   The following example shows the execution plan panels for CompositeView.



3. Highlight the view name in the Tree Panel to see details for the overall query in the Details Panel.
4. Highlight a node (SELECT, one of the JOINs, and so on in this example) in the Tree Panel to see details for that node in the Details Panel.

To display the SQL execution plan in Studio Manager

1. Open Studio Manager by clicking the middle vertical tab along the left side of the Studio window.
2. Select Requests in the navigation pane on the left.
3. Select an SQL query from the list in the Name column.

   When you select a query, the Cancel and Show Query Plan buttons become available.

   You can expand the Name column to show the first 50 characters of each query text to help identify the query of interest.
4. Click Show Query Plan near the center of the window.

   When you highlight a node in the tree panel on the left side, information is provided for that element as described in Execution Plan Nodes in the Tree Panel, page 555.

Execution Plan Nodes in the Tree Panel

The following table lists the nodes that can appear in the Tree Panel. This panel is displayed at the bottom left of the Studio pane when you click Show Execution Plan. The tree is a hierarchical representation of the query plan.

Note: These nodes are also listed in the logged execution plan.

When you select one of these nodes, you see a list of Execution Plan Query Attributes in the Details Panel, page 558.

Node	Functionality
AGGREGATION	Shows plans for aggregate functions that are not part of a GROUP BY clause.
CROSS JOIN	Merges two streams of incoming rows and produces one stream of rows that is the Cartesian product of the two streams.

Node	Functionality

Pre Data Ship Plan	Represents the query execution analysis prior to the shipment of one or multiple nodes to the data ship target. After a data ship is executed, each of the Pre Data Ship Plan nodes state:
	`Execution Status:`
	`NOT EXECUTED. This operation was determined unnecessary or was cancelled before any rows were retrieved.`
	
	The Pre Data Ship Plan nodes are not executed because the query is reworked to execute the operation on the data ship target.

DISTINCT	Removes all incoming duplicate rows.
FETCH	Produces the rows resulting from execution of a query on a data source. The information that can be displayed includes: • Estimated rows returned (or "Unknown") • Data source path, type, and driver name • Data ship target, if data ship is being used • Data ship notes, if any • SQL of the fetch • SAP BW OLAP View runtime notes with estimated rows returned The FETCH node and the SQL submitted to the data ship target reveals how a query that uses data shipment is rewritten to select data that was shipped into a local temp table to make it available for a collocated operation compared to the operation that was evaluated in the Pre Data Ship Plan.
FILTER	Passes through only the incoming rows that satisfy the filter criterion.
FULL OUTER JOIN	Merges two streams of incoming rows and produces one stream containing the SQL FULL OUTER JOIN of the two streams. Refer to SQL reference materials for a description of FULL OUTER JOIN.
FUNCTION	Shows how a function is executed.
GROUP BY	Reorders the incoming rows so that they are grouped by some criterion. For example, if the rows are grouped by a name, all rows with the same name are combined into a single row.
IN SUBQUERY	Lists a node for an unpushed IN subqueries.

Node	Functionality
JOIN	Merges two streams of incoming rows and produces one stream containing rows that satisfy a criterion that applies to both streams. The information displayed includes: • Estimated rows returned (or "Unknown") • Criterion applied • Algorithm used • Algorithm notes • Estimated left and right cardinality
ORDER BY	Reorders the incoming rows to satisfy a sorting criterion.
PROCEDURE	Produces rows resulting in the execution of a query or stored procedure call on a data source. The information displayed includes: • Estimated rows returned (or "Unknown") • Location of the SQL statement • Reason for not pushing if that is the case
RIGHT OUTER JOIN	A right outer join performed as part of a stored procedure. Merges two streams of incoming rows and produces one stream containing the SQL right-outer-join of the two streams. Refer to SQL reference materials for a description of RIGHT OUTER JOIN.
SELECT	Applies functions on the column values of the rows. This node produces the same number of rows that it reads. The information displayed includes: • Estimated rows returned (or "Unknown") • Projection of the SELECT statement • Data ship notes, if any
UNION	Combines two streams of incoming rows and produces a single stream. The cardinality of produced rows equals the sum of the cardinality of the incoming streams. The order in which the node produces rows is undefined.
CROSS PROCEDURE JOIN	A cross-join performed as part of a stored procedure.

Node	Functionality
FULL OUTER PROCEDURE JOIN	A full outer join performed as part of a stored procedure.
FETCH WITH	Same as FETCH, but containing a specified condition.
UPDATE	Updates a data source, typically a table.
INSERT	Inserts data into a data source, typically a table.
DELETE	Deletes data from data source, typically a table.
MERGE	Merges data into a data source, typically a table. Merge inserts a row if it is not present, and updates a row if it already exists.
WITH SUBQUERIES	Subqueries that are part of WITH clauses.
INTERSECT	
EXCEPT	

Execution Plan Query Attributes in the Details Panel

The following table lists attributes that can appear in the Details Panel when the overall query (request) is highlighted in the Tree Panel. This panel is displayed at the bottom right of the Studio Modeler pane when you click Show Execution Plan, or of the Studio Manager pane when you click Show Query Plan.

These attributes can also be found in the logged execution plan, although some of them are estimates *before* execution and actual run-time values *after* execution.

Field	Description
Background data source read time	Time spent by background threads in all FETCH, page 556 and PROCEDURE, page 557 nodes in the execution plan.
Background server processing time	Time spent by background threads in all the nodes (except for FETCH, page 556 and PROCEDURE, page 557) in the execution plan.
Data ship data transfer time	Time required to transfer data from one data source to another.

Field	Description
Elapsed execution time	Amount of wall-clock time that the server used to execute the query. This time is the total of Query initialization time, page 559, Foreground server processing time, page 559, and Foreground data source read time, page 561.
Foreground server processing time	Fraction of the Elapsed execution time, page 559 that the server used in the actual execution of the query; that is, the processing time of the nodes in the execution plan. This time does not include the time used to read rows from the data sources. By comparing this time with Foreground data source read time, page 561, you can determine how much time was spent by the server versus the time spent in the data sources.
Peak memory reserved	Peak memory reserved for current request.
Query initialization time	Time the server used to analyze the query, create and optimize an execution plan, rewrite the query, and establish connections (from the pool if pooling is configured) to the data sources. If data ship is involved, time is spent creating a data ship temp table at the data ship target, and shipping the data. With multiple data shipments to multiple temp tables, these tasks are done in parallel by separate threads, and the shipped segment that takes the longest keeps the query initialization time clock running.
Reset count	
Resolved SQL	Fully resolved SQL text.
Rows modified	Number of rows modified by the request.
Rows returned	Number of rows produced by an execution node. If you want to know how many rows were read by the node, look at the returned row counts of the node's children.
Server version	Version of the server software.
Speed up due to concurrency	Estimate of how much faster the query ran because of threading. For example, 100% means the query ran twice as fast with threading.
Submitted SQL	Original SQL text.

Execution Plan Node Attributes in the Details Panel

The following table lists attributes that can appear in the Details Panel. This panel is displayed at the bottom right of the Studio Modeler pane when you click Show Execution Plan, or of the Studio Manager pane when you click Show Query Plan. When you select a node in the Tree Panel on the left, its attributes are displayed.

The following table lists attributes that can appear in the Details Panel when a node is highlighted in the Tree Panel. This panel is displayed at the bottom right of the Studio Modeler pane when you click Show Execution Plan, or of the Studio Manager pane when you click Show Query Plan.

Note: See Execution Plan Nodes in the Tree Panel, page 555, for a list of nodes.

These attributes can also be found in the logged execution plan, although some of them are estimates *before* execution and actual run-time values *after* execution.

Field	Description
Algorithm	Name of the algorithm used by the node; for example HASH JOIN or SORT/MERGE JOIN.
Algorithm notes	More details about the algorithm used.
Background data source read time	For SCAN or PROCEDURE: how much time the child thread spent on reading from the data source.
Background node processing time	For nodes other than SCAN, PROCEDURE, INSERT, UPDATE, DELETE, or MERGE: time spent processing this node by a background thread. This time is zero if the node was processed by a **foreground** thread.
Base DN	For SCAN.
Candidate keys	For SELECT.
Connection ID	For SCAN.
Criteria	Shows the criteria used by a JOIN, FILTER, GROUP BY, or ORDER BY node. Condition or predicate applied during an operation
Data ship notes	For SCAN or SELECT. A report of the estimated number of rows to be shipped, and the amount of time consumed to get the cost estimate. When a condition blocks the use of data ship, this field usually describes the condition, case, or option that blocked optimization.
Data ship SQL	For SCAN.

Field	Description
Data ship target	For SCAN. Shows where the results of SQL execution will be sent. The presence of Data ship target indicates that data ship will be performed.
Data ship transfer time	For SCAN.
Data source driver name	For SCAN or PROCEDURE.
Data source path	For SCAN or PROCEDURE.
Data source type	For SCAN or PROCEDURE.
Data source write time	For INSERT, UPDATE, DELETE or MERGE.
Estimated left cardinality	Estimated cardinality for the left input in a binary join. Pre-execution (Show Execution Plan button) Details Panel only
Estimated right cardinality	Estimated cardinality for the right input in a binary join. Pre-execution (Show Execution Plan button) Details Panel only
Estimated rows returned	Pre-execution (Show Execution Plan button) Details Panel only. Logged execution plan records actual Rows Returned.
Execution status	If the operation for the node is still running, status can be PARTIAL RESULTS or NOT EXECUTED.
Fetch rows	For SELECT.
Foreground data source read time	For SCAN or PROCEDURE. Fraction of the Elapsed execution time, page 559 that the main thread used to read data from the data sources. By comparing this time with Foreground server processing time, page 559, you can determine how much time was spent by the server vs. the time spent in the data sources.
Foreground node processing time	For nodes other than SCAN, PROCEDURE, INSERT, UPDATE, DELETE, or MERGE: fraction of the elapsed time used by the node. This time is zero if the node was processed by a background thread.
Name	
No push reason	Why the node was not pushed to the data source.
Offset rows	For SELECT.

Field	Description
Peak memory reserved	Peak memory reserved for current node.
Peak memory used	Peak memory used for current node.
Procedure join number of invokes	For PROCEDURE.
Procedure type	For PROCEDURE.
Projection	Set of output columns in a SELECT node.
Rows deleted	
Rows inserted	
Rows merged	
Rows returned	
Rows updated	
Runtime notes	
Semi-join criteria	For SCAN.
Semi-join partition count	For SCAN.
SQL	SQL text executed by the node. In a FETCH, page 556 or PROCEDURE, page 557 node, this field contains the actual data-source-specific query that was pushed.
UniqueIndex	For SELECT.

Updating the Query Execution Plan

An execution plan is generated the first time a SQL statement is executed, and whenever the query changes. The query execution plan is cached, so if the underlying data or statistics change, it is a good idea to regenerate the execution plan as described below.

A simple way to force a refresh of an execution plan is to add a space to the query. You can also flush the query plan cache for all views by temporarily toggling the Query Plan Cache Enabled key in the TDV Configuration window as described in Refreshing All Execution Plan Caches, page 565.

To update the query execution plan

1. If statistics have changed on the view, gather statistics as described in Creating Cardinality Statistics for a View, page 566.
2. Regenerate the execution plan using one of these methods:
   - Click Show Execution Plan on the view editor toolbar. (For an OLAP View editor, the button is on the toolbar of the SQL tab.

   Or

   - In the Execution Plan panel, click Generate Plan.
3. Optionally, type a value in the Refresh Rate field, to specify a regular interval at which to update the execution plan.

Viewing How Much Data was Processed by a Query

You can execute a query and determine how much data was processed by each node, and which nodes took the most time in the query execution plan.

Query plans in SQL that call procedures in FROM clauses show the details of what the procedures do (in the PROCEDURE nodes) when you direct the tool to execute and show the statistics.

If Keep Result is selected, the results of the query execution are available in a new Result tab. The data in the tab is available for as long as you keep the resource open in Studio. The query execution ID appears both at the top of the Execution Plan nodes in the left pane and in the Result tab name to help you match statistics and query information to results.

To view how much data was processed by a node

1. Generate an execution plan.
2. In the Execution Plan panel, select a node in the left pane.
3. Optionally, select Keep Result in the lower right corner of the Execution Plan to view the results of this query in a separate tab.
4. Click Execute and Show Statistics.

   Statistics are added to the nodes in the left pane of the Execution Plan.

Each node name is followed by values in parentheses like (*n*) or (*n*, *m*%), where *n* is the number of rows produced by that node, and *m* is the percentage of elapsed time that the node used to process the data. If *m* is not shown, it is 0.

For example, if the elapsed time was 60 seconds and *m* is 20, the node accounted for 12 seconds of the elapsed time (20 x 60). If *m* is 0, processing at the node did not contribute to elapsed time—for example, if the node was processed by a background thread and its processing was completed before the rows were needed by the parent node. The *m* percentages help determine which nodes to focus on to improve performance.

5. Optionally, set a refresh rate for the execution plan.
6. Optionally, click Refresh Now during the execution of the query.

   This updates the statistics being gathered on data sources and tables, which can take a long time for long-running queries.

For Example

```
Name:
SELECT
Rows Returned:
220
Estimated Rows Returned:
Unknown
Total execute time which include children time and wait time.:
121.4 msecs
Foreground Node Processing Time:
2.5 msecs
Peak memory reserved:
4000000
Projection:
orderdetails0.orderid OrderID, orderdetails0.productid ProductID,
orderdetails0.discount Discount, orders.orderdate OrderDate,
customers.companyname CompanyName, customers.contactfirstname
CustomerContactFirstName, customers.contactlastname
CustomerContactLastName, customers.phonenumber
CustomerContactPhone, productCatalog_xform.ProductName
ProductName, inventorytransactions.transactionid TransactionID,
purchaseorders.daterequired DateRequired,
purchaseorders.datepromised DatePromised, purchaseorders.shipdate
ShipDate, suppliers.supplierid SupplierID, suppliers.suppliername
SupplierName, suppliers.contactname SupplierContactName,
suppliers.phonenumber SupplierPhoneNumber
Data Ship Notes:
Data Ship Query is not possible. No suitable source scan or ship-to
target found.
```

Refreshing All Execution Plan Caches

TDV Server caches view-query execution plans. If the view SQL uses data sources with fresh cardinality statistics, you might want to purge and refresh the query plan cache by temporarily changing the Query Plan Cache configuration setting, which forces a one-time regeneration of all execution plans.

To purge all query execution plans

1. Select the Administration > Configuration menu option.
2. In the Configuration window, select TDV Server > SQL Engine > Caches > Query Plan Cache Enabled.
3. In the right pane, select the False radio button and click Apply.
4. Select the True radio button, click Apply, and click OK.

   Or

   Refresh a query plan cache by making a minor change to the View SQL or procedure. This forces creation of a new query execution plan.
5. Save the change.

   The next execution uses any cardinality statistics available to optimize query plan execution.

Creating Cardinality Statistics for Cost-Based Optimization

The TDV SQL Query Engine might use statistics gathered from the data sources, when they are available, to create an efficient SQL query execution plan for joins or unions across tables, or to optimize caching. Statistics gathered on tables or views provide the SQL Query Engine with estimates on the table cardinality (the number of unique values or rows) so that optimizations can be applied.

By default, no statistics are gathered for any data source, and statistics storage tables must be created explicitly at the level of the data source to enable selected table and column scans.

Statistical data associated with a data source can be exported and imported with the data source, or you can just back up the configurations.

Any user with Access Tools and Modify All Resources rights, and with READ and WRITE privileges on a data source, can set up the gathering of cardinality statistics on the data source and one or more of its tables.

This section includes:

- Creating Cardinality Statistics for a View, page 566
- Creating Cardinality Statistics on a Data Source, page 568

You can consult the system data tables (/services/databases/system/SYS_STATISTICS) to see the status and the metadata (if you have the Read All Resources right) from both the data source and individual tables.

Creating Cardinality Statistics for a View

If you intend to use the view for caching, you can collect cardinality statistics for it.

Note: You must set up the cache before you define the cardinality statistics gathering profile. See Setting Up Caching, page 494 for more information.

For the TDV parallel option, you must gather statistics for the column that is the numeric simple key for the view. This column should be either the one identified as the primary key column by the execution plan, or the primary key that you have defined using the Indexes tab.

To collect cardinality statistics for a view

1. From the resource tree, open the view for which you want to collect cardinality statistics.
2. Select the Caching tab.
3. Click Create Cache.

   For more information on creating a cache, see Caching to a File Target, page 496.
4. If required by Studio, define the Cardinality Statistics for the data source. For more information, see Creating Cardinality Statistics on a Data Source, page 568.
5. Select the Cardinality Statistics tab for the view.
6. Click Create Statistics.
7. Check Enable to allow use of the gathered table statistics to optimize execution plans.

   You do not need to check the Enable check box to *configure* statistics gathering, but you do need to check that box for the optimizer to *use* the statistics. To disable statistics gathering and use of existing statistics, clear the Enable check box and save.

8. Optionally, define cardinality override settings:
   - Minimum Cardinality—The minimum number of returned rows you would expect from a SQL SELECT on this table. Zero (0) is a valid number.
   - Maximum Cardinality—The maximum number of rows the resource could return.
   - Expected Cardinality—Typical number of unique rows returned from this table.
9. Specify the schedule for collecting cardinality statistics: On Cache Refresh or Manual. You can use the Gather Now button at any time.
10. Optional. Set the Statistics Gathering Timeout value.

    When the table statistics gathering timeout is set to "-1" the data source statistics gathering timeout is used. A value of 0 (the default) indicates that there is no time limit.

    If the statistics processing is already returning data from a full table scan, the timeout setting does not stop processing immediately; instead, the timeout triggers an attempt to make statistical estimates based on the subset of data already returned.
11. Specify your table statistics gathering preferences:
    - None—Do not gather statistics for the table.
    - Use datasource settings—Gather statistics according to the definitions specified for this data source.
    - Gather table boundary statistics—Performs SELECT COUNT (*) on each specified table to count its rows.
    - Specify gathering for specific columns—Control statistics gathering column by column.
12. Optionally, select the Specify gathering for specific columns option and then specify column gathering rules and manipulate the values for MIN, MAX, and distinct.
    a. In the Gathering Rule column, double-click to select one of the first two options in the table below to gather statistics.

Gathering Rule Option	Choose to...
Gather all statistics	Collects all other statistics plus a full table scan of numeric data types to build a histogram. Or, builds a full string index for string data types. BLOB, CLOB and other data types that yield meaningless statistical data is not configurable for statistical evaluation.

Gathering Rule Option	Choose to...
Gather boundary statistics	Collect table boundary statistics and gather minimum and maximum, and count distinct values for numeric data types. For string data types, calculate the number of distinct values for tables with fewer than ten thousand rows. To reduce the cost of gathering statistics for column boundaries, apply one or more assumptions.
Do not gather statistics	Stop gathering statistics on the column (default).

b. Use one or more of the following fields to override and avoid processing of statistics in the database.

This can be a valuable way to avoid gathering column boundary statistics and have TDV use the values you specify when the queries are run.

When TDV gathers statistics, it retrieves all data from the source and builds a histogram and string index based on this data. This operation can be expensive, depending on the number of rows and columns, network bandwidth, and so on. If you have limited time for statistics gathering, overriding the values can be useful.

These column values are used to create a uniform distribution at run time for the evaluation of the selections. If a value is specified for a column, statistics are not gathered for that column.

— Minimum Value—An option to override the minimum value for the column.

— Maximum Value—An option to override the maximum value for the column.

— Distinct Count—An option to override the number of distinct values for the column.

— Num of Buckets—An option to increase or decrease the granularity of the statistics.

13. Save your selections.

Creating Cardinality Statistics on a Data Source

Statistics gathering must be enabled on the parent data source. You create statistics from the Cardinality Statistics tab, which is available for any data source.

Note: The data source sign-in and password must be saved to enable statistics gathering.

To collect statistics on a data source

1. From the resource tree, open a data source.
2. Select the Cardinality Statistics tab.
3. Click Create Statistics.
4. Check Enable to allow use of the table statistics gathered to optimize execution plans.

   You do not need to check the Enable check box to configure statistics gathering, but you need to check that box for the optimizer to use the statistics. To disable statistics gathering and use of existing statistics, clear the Enable check box and save.
5. For Mode, select one of the following:
   - Do not gather statistics—The default setting for all data sources and tables.
   - Gather table boundary statistics—Counts the number of rows in tables, performing a SELECT COUNT (*) on each of the specified tables.
   - Gather column boundary statistics—Gets table boundary statistics and gathers the minimum and maximum, and counts distinct values for numeric data types. For string data types, calculates the number of distinct values for tables with fewer than ten thousand rows. The cost of gathering statistics for the column boundary statistics is reduced by application of one or more assumptions.
   - Gather all statistics—Collects all the above, plus a full table scan of numeric data types, to build a histogram. For string data types, also builds a string index.
6. Optionally, set the Number of Threads.

   Number of Threads specifies the number of low-priority, query-executing threads allowed on the database and on TDV.

   Note: Some databases do not tolerate large numbers of threads scanning tables at the same time. TDV also has a thread capacity limit that can cause disk paging when nearing memory thresholds. You can specify a maximum number of threads (without needing a server restart) by selecting Administration > Configuration, and in that window navigating to Server > SQL Engine > Optimizations and setting Max Number of Statistics Gathering Threads In The System to whatever value works in your environment.

7. Set the statistics gathering schedule:
   - Manual—Use Gather Now to gather statistics right away. If your cache is controlled by a cache policy, it is probably better to use a periodic or programmatic refresh.
   - Exactly Once—Requires that you set a day and time in the future to run.
   - Periodic—Statistical collection occurs at timed intervals, such as at an off-peak hour each night.

   Frequency of statistics gathering also depends on data volatility. Some tables never change and others change frequently. Gather statistics based on the expected frequency of cardinality and table boundary change.

   It is best to use off-peak hours for gathering data source statistics, a process that can use significant resources. You can use multiple threads for data sources that support parallel processing.
8. In the Time-out in minutes field, specify the number of minutes permitted for gathering data from a resource.

   A value of 0 or null (the default) indicates that there is no time limit.

   If the timeout expires after statistics processing is already returning data, the timeout triggers an attempt to make statistical estimates from the subset of data already returned.
9. Save your selections.
10. Define cardinality statistics gathering option for the table or view that you are interested in by following the steps in Creating Cardinality Statistics for a View, page 566.

Scheduling Cardinality Statistics Refresh

You can schedule statistics gathering to happen at the best time for you. Use the following procedure, or refer to the steps in Using TDV API Procedures to Refresh or Cancel Resource Statistics, page 572.

To schedule refreshes of the cardinality statistics

1. Follow the instructions in Creating Cardinality Statistics for a View, page 566 or Creating Cardinality Statistics on a Data Source, page 568.
2. Set the statistics gathering schedules for one of the following:
   - Manual—Use Gather Now for immediate collection of statistics. If your cache is controlled by a cache policy, use a periodic or programmatic refresh.
   - Exactly Once—Requires that you set a day and time in the future to run.
   - Periodic—Statistics collection occurs at timed intervals, such as at an off-peak hour each night.

   Frequency of statistics gathering also depends on data volatility. Some tables never change and others change frequently. Gather statistics based on the expected frequency of cardinality and table boundary change.

   It is best to use off-peak hours for gathering data source statistics. You can use multiple threads for data sources that support parallel processing.
3. Use the Stats Gathering Timeout field to specify the number of minutes permitted for gathering data from a resource.

   A value of 0 (zero; the default) indicates that there is no time limit.

   If the timeout expires when statistics gathering is already returning data, the timeout triggers an attempt to make statistical estimates from the subset of data already returned.
4. Save your selections.

Refreshing Cardinality Statistics

You can schedule statistics gathering and refreshing for a time that is best for you. You can use the following procedure, or refer to the steps in Using TDV API Procedures to Refresh or Cancel Resource Statistics, page 572.

To refresh cardinality statistics

1. Follow the instruction in Creating Cardinality Statistics for a View, page 566 or Creating Cardinality Statistics on a Data Source, page 568.
2. Set the statistics gathering schedules for one of the following:
   - Manual—Use Gather Now to begin collecting statistics immediately. If your cache is controlled by a cache policy, use a periodic or programmatic refresh.
   - Exactly Once—Requires that you set a day and time in the future to run.
   - Periodic—Statistics collection occurs at timed intervals, such as at an off-peak hour each night.

   Frequency of statistics gathering also depends on data volatility. Some tables never change and others change frequently. Gather statistics based on the expected frequency of changes to cardinality or table boundaries.

   It is best to use off-peak hours for gathering data source statistics and to use multiple threads for data sources that support them.
3. Use the Stats Gathering Timeout field to specify the number of minutes permitted for gathering data from a resource.

   A value of 0 (zero; the default) indicates that there is no time limit.

   If the timeout expires but statistics processing is already returning data, the timeout triggers an attempt to make statistical estimates from the subset of data already returned.
4. Save your selections.

Using TDV API Procedures to Refresh or Cancel Resource Statistics

The following built-in procedures are available for using resource statistics:

- localhost/lib/resource/RefreshResourceStatistics
- localhost/lib/resource/CancelResourceStatistics
- services/databases/system/SYS_STATISTICS

To view a detailed procedure description

1. Double-click the procedure to open it.
2. Click the Info tab to display the procedure annotation.

Use Indexes, and Primary and Foreign Keys, to Improve Query Performance

Indexes can speed up searching or retrieving data from a database table. Identifying indexes and primary keys enable the TDV Query Engine to use logical algorithms for faster, more efficient joins.

Typically, when an index is created for a column, all data in that column is scanned and a data structure (index) is constructed; the index makes the data in that column faster to lookup, although indexing also can slow transactions that insert rows into the tables. The contents of the index are automatically updated whenever the data in a row are modified.

In Studio, the Indexes panel in the view and table editor lets you create metadata labels for indexes and primary keys already present in the data source. In Studio, the Foreign Keys panel in the view and table editor lets you create a definition that uses any foreign keys present in the data source.

TDV does not build a specified data structure, but creates a metadata definition that lets you take advantage of the existing resource's data structure. Queries and joins do not necessarily run faster because of an index. The metadata index and key specifications are not validated, so there is no guarantee that a column marked as a primary key has unique values.

Clients of the TDV Server can benefit from knowledge of indexes and primary keys feature because this information is made available when the data service is published. More sophisticated client programs can generate efficient queries based on column indexes and primary keys.

In some cases, TDV cannot automatically determine what columns of a view are indexed. For example, a column might be a primary key in the base table. But in the view, the value might not qualify as a primary key.

For information on how to create an index for a table or view in Studio, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.

Types of Joins

Inner and outer joins can use an equality condition (equijoins) or not (non-equi-joins). Most table joins are equi-joins, comparing columns and rows with like values. The nested loop algorithm is used for non-equi-joins.

The following table lists algorithms used and the types of joins that they can be used with.

Algorithm	Inner equi-join	Non-equi-join	Left/right outer join	Full outer join	Semijoin optimization
Hash	yes		yes	yes	yes
Nested Loop	yes	yes	yes	yes	yes
Sort Merge	yes		yes		

TDV uses several algorithms to execute joins on table columns. By default, the TDV query engine attempts to find the best algorithm and join optimization to use for your SQL if none is specified. The Join Properties editor offers the following join algorithm options:

- Automatic Option, page 574
- Hash Join Option, page 575
- NESTEDLOOP Join Option, page 575
- SORTMERGE Join Option, page 575
- Semijoin Optimization Option, page 579
- Star Schema Semijoin, page 587

Automatic Option

By default, the TDV system automatically optimizes the SQL execution plans for any query, given estimates of left and right cardinality and other statistical information. Automatic is not a join algorithm or a join optimization, but it lets the TDV query engine optimize based on an analysis of the SQL using known database statistics.

You can specify the SQL execution plan explicitly with one of the other options; however, specification of a join option only influences what join algorithm or optimization is first attempted, not necessarily the one ultimately used. If the option specification is incompatible with the data sources or incorrectly specified, the execution plan might change to a compatible join that was not specified.

Hash Join Option

The HASH algorithm is useful when you are joining tables with large amounts of data. The optimizer uses the inner table to build a hash table on the join key in memory. It then scans the outer table, probing the hash table to find the joined rows.

The HASH algorithm requires only a single pass on each table. It is best used when the inner table fits into available memory.

However, if the hash table does not fit into available memory, the optimizer breaks it into partitions that are written to temporary segments on disk.

If the process is expected to write to disk, you can use the FORCE_DISK query hint (described in the *TDV Reference Guide*) to make the process write to disk from the beginning, saving read/write cycles and excessive memory usage.

NESTEDLOOP Join Option

You can use the NESTEDLOOP algorithm without any data structure information, but it is not inherently optimized for performance. NESTEDLOOP is useful when small subsets of data are joined and the join condition is an efficient way to access the right table.

When the outer (driving) table has numerous rows that force multiple probes of the inner table, the NESTEDLOOP algorithm becomes costly to run.

SORTMERGE Join Option

The SORTMERGE algorithm is a streaming operator. If both input streams can be ordered given the join criteria and compatibility, the join operator works efficiently and quickly in a small memory footprint.

SORTMERGE can also impose other requirements, such as both sides of the join being ordered. To meet that requirement, ordering must be compatible with the join criteria. For example, consider the following join criteria:

```
ON T1.A = T2.X AND T1.B = T2.Y
```

To perform the SORTMERGE join, the left side has to be put into ascending order by A and B, and the right side has to be put into ascending order by X and Y.

If a side is not ordered or does not have compatible ordering, the query engine automatically checks whether a compatible ORDER BY can be pushed to the data source so that the SORTMERGE join can proceed.

SORTMERGE is not compatible with the SQL when ORDER BY cannot be pushed to the data sources. Ordering must be performed on both sides for a SORTMERGE join.

To prevent the query engine from choosing SORTMERGE over HASH, specify the value of the SORTMERGE option as false, as follows:

```
{option SORTMERGE="false"}
```

SQL Join Reordering

TDV automatically assesses SQL Join blocks (inner joins and all types of outer joins) to determine suitability for rewriting the SQL join order. A Join block is a set of contiguous joins. The TDV query engine analyzes each join block of the query independently, and attempts to rewrite the query to maximize the number of joins pushed down to data sources. While inner joins can be reassociated without restriction, the TDV query optimizer checks a number of technical rules to determine if it is safe to reorder a block that contains outer joins.

For queries that meet complex criteria, TDV rewrites the SQL join order if it expects that to yield a performance gain. Joins are reordered so that tables from the same data source are performed first, which maximizes push down and makes the query more efficient.

The following examples show how SQL join ordering works.

```
SELECT O10BIN_ID, O100BIN_ID, O1KBIN_ID
FROM /users/composite/test/sources/oracleA/QABIN/O10BIN RIGHT
OUTER JOIN
        /users/composite/test/sources/oracle/QABIN/O100BIN
    ON O10BIN_NUMBER1 = O100BIN_NUMBER1 INNER JOIN
        /users/composite/test/sources/oracle/QABIN/O1KBIN
    ON O100BIN_FLOAT = O1KBIN_FLOAT
```

The query as written would perform first the outer join and then the inner join, because by default joins are left-associative. The plan for this query looks like this:

```
[1] Request #2511
[2] + SELECT   (13)
[3]   + JOIN   (13)
[4]     + RIGHT OUTER JOIN   (100)
[5]     | + FETCH   (10) [Oracle2]
[6]     | + FETCH   (100) [Oracle]
[7]     + FETCH   (1000) [Oracle ]
```

The numbers in parentheses represent number of rows produced by each operator. The annotations in square brackets are data source names.

The same query could be rewritten like this:

```
SELECT O10BIN_ID, O100BIN_ID, O1KBIN_ID
FROM /users/composite/test/sources/oracle2/QABIN/O10BIN RIGHT
OUTER JOIN
        (/users/composite/test/sources/oracle/QABIN/O100BIN INNER
JOIN
            /users/composite/test/sources/oracle/QABIN/O1KBIN
```

```
ON O100BIN_FLOAT = O1KBIN_FLOAT)
ON O10BIN_NUMBER1 = O100BIN_NUMBER1
```

The join between the two tables from the same data source is performed first, and the optimizer is able to push down this join to the data sources. The corresponding query plan looks like this:

```
[1] Request #2533
[2] + SELECT   (13)
[3]   + RIGHT OUTER JOIN   (13, 9%)
[4]     + FETCH   (10, 36%) [Oracle 2]
[5]     + FETCH   (13, 59%) [Oracle]
```

TDV has to perform only one join. Most of the data is filtered out at the data source, reducing data transfer. TDV performs automatically rewrites join blocks that include inner joins and all types of outer joins.

A join block is a set of contiguous joins. The query engine analyzes each join block of the query independently and attempts to rewrite the query to maximize the number of joins pushed down to data sources. While inner joins can be reassociated without restriction, the optimizer checks a number of technical rules to determine if it is safe to reorder a block that contain outer joins. Cross-joins are not eligible for reordering. Join reordering is not performed if it introduces new cross joins. A join node with hints is not considered for reordering.

A precondition of the join ordering algorithm is that all outer joins can be simplified. Here is an example:

```
SELECT O10BIN_ID, O100BIN_ID, O1KBIN_ID
FROM /users/composite/test/sources/oracle2/QABIN/O10BIN LEFT OUTER
JOIN
        /users/composite/test/sources/oracle/QABIN/O100BIN
    ON O10BIN_NUMBER1 = O100BIN_NUMBER1 INNER {OPTION SEMIJOIN}
JOIN
        /users/composite/test/sources/oracle/QABIN/O1KBIN
    ON O100BIN_FLOAT = O1KBIN_FLOAT
```

The left outer join can be converted to an inner join because the inner join above it has a null-rejecting predicate that applies to the null-filled side of the outer join. The ON clause of the top join rejects any rows produced by the left outer join that have NULL for the O100BIN_FLOAT column.

The plan for this query looks like this:

```
[1] Request #3040
[2] + SELECT   (2)
[3]   + JOIN   (2)
[4]     + JOIN   (1)
[5]     | + FETCH   (10) [Oracle2]
[6]     | + FETCH   (89) [Oracle]
[7]     + FETCH   (2) [Oracle]
```

This shows that left outer join was converted to inner join but that join reordering did not take place because the join hint implies that the inner join (in the input query) cannot be reordered.

TDV performs the SQL join reordering analysis automatically. Special configurations or SQL options are needed only if you want to enforce SQL processing as it is written because you want table construction or table relationships.

You can join the tables in any order you want. (See Views and Table Resources, page 215.)

Enforce Join Ordering

You can direct the query engine to follow the order in which you joined the tables. For information on how to specify joins, see Views and Table Resources, page 215.

Multiple Join Conditions

If the source side has many rows and produces a prohibitively long SQL string on the target side, the TDV query engine deals with it in one of these ways:

- Partitioned semijoin—The TDV query engine breaks up the target-side predicate into multiple chunks, and submit one query for each chunk to the target side. This technique applies regardless of whether the ON clause uses conjunctions or is just a single predicate, and regardless of whether the target side SQL is using the IN clause or OR syntax. The partitioned semijoin is also applied to semijoin queries with non-equality conditions.
- Query predicate rewriting—If a join has conjunctions in the ON clause and the target-side data source does not support the row-constructor form of the IN clause, then instead of generating a target side row-constructor form using an IN clause like:

```
(t1,t2) IN ((1,2), (6,3), (8,4),(93,3)…)
```

  The TDV Server might generate a predicate with two or more scalar IN clauses like this:

```
t1 IN (1,6,8,93") and t2 IN (2,3,4,3,…).
```

  The scalar IN clauses syntax is more compact than the OR syntax, and it reduces the chance of partitioned semijoin and load on the target data source. However, this predicate is not as selective as the OR form, so it could bring more rows into TDV. Because of this trade-off, you can configure the number of rows on the source side that causes the engine to switch from the OR syntax to multiple scalar IN clause syntax.

Consider this more complicated example:

```
SELECT * from DS1.R LEFT OUTER JOIN DS2.T on R.r1 = T.t1 and R.r2 =
T.t2
```

In this case, the TDV Server query engine might generate an additional predicate (the row-constructor form of the IN clause) on the T-side that takes the form:

```
(t1,t2) IN ((1,2), (6,3), (8,4), (93,3)…)
```

If the DS2 target supports this form of IN clause query phrasing, it is preferred.

If the data source does not support this form of IN clause, the additional predicate looks like this:

```
(t1=1 and t2=2) or (t1=6 and t2=3) or (t1=8 and t2=4) or …)
```

The second form is logically identical to the IN-clause form, but if the right side produces many rows, it could generate a long SQL string on the target side, potentially exceeding the capabilities of DS2.

- Pushing left outer joins with multiple predicates—If the query engine can predetermine what is selectable based on an ON clause and a WHERE clause, it pushes the query to the data source. For example:

```
SELECT
O10_Id
O10_CHAR
S_Id
S_CHAR
FROM /users/eg/t1 left outer { option semijoin="false", hash } join
/users/sqlsr/5kk
ON O10_Id = S_Id
WHERE O10_Id = 9
```

Here, the WHERE clause predicate is the same as the ON clause predicate, so O10_Id = 9 can be pushed to the data source, which can dramatically reduce the amount of data returned.

Semijoin Optimization Option

A semijoin between two tables returns rows from the first table whenever one or more matches are found in the second table. The difference between a semijoin and a conventional join is that rows in the first table are returned at most only once, even if the second table contains two matches for a row in the first table.

This section contains:

About the Semijoin Optimization

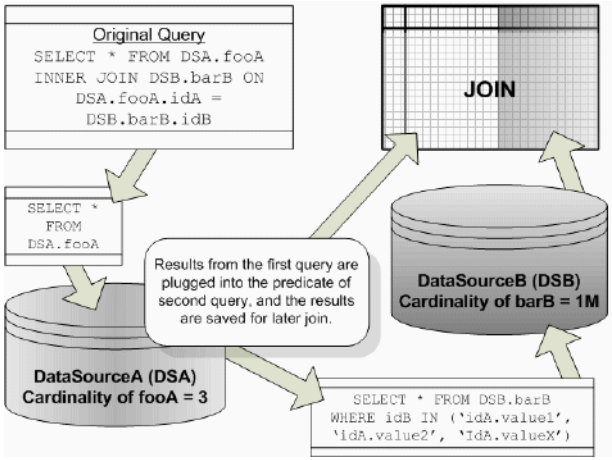
The semijoin optimization reduces the number of rows retrieved from the right-hand side (RHS) by rewriting the FETCH pushed to the second data source using the unique values returned by the left-hand side (LHS). Table cardinalities must be supplied for this optimization.

Inner joins can use the semijoin optimization. Left and right outer joins can use semijoin optimization, but only from the outer side to the inner side. The full outer join is not supported with the semijoin optimization.

You can specify use of the semijoin optimization in the SQL defining the view. When the table cardinalities of source and target sides are known, and when the ratio between the two is favorable, the TDV query engine automatically uses the semijoin optimization.

The semijoin can only be attempted if the right side can be queried as a node that fetches against a data source that supports an IN or an OR clause.

For inner and left outer joins, the semijoin optimization always uses rows from the LHS to constrain the RHS—for example, where LHS is the source side and RHS is the target. For a right outer join the situation is reversed: RHS is the source and LHS is the target.



As another example, consider the tables Employee and Dept and their semijoin.

Employee Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production

Dept DeptName	Manager
Sales	Bob
Sales	Thomas
Production	Katie
Production	Mark

The semijoin of the employee and department tables would result in the

Employee joined to Dept Name	EmpId	DeptName
Sally	2241	Sales
Harriet	2202	Production

following table.

Consider another query like this:

```
SELECT * FROM DS1.R INNER JOIN DS2.T ON R.r = T.t
```

The usual evaluation would hash the R table expression and then iterate over T table expressions, and do look-ups into the hash table to find matching rows. If the join in the example is such that the number of rows from the R-side is much smaller than the number of rows from the T-side, additional optimization is possible. The rows from R are buffered in TDV, but an additional predicate of the form "t IN (1,6,8,93…)" is added to the SQL generated for the RHS. The values on RHS of the IN clause are all the 'r' values from LHS of the join. This additional predicate ensures that TDV retrieves the smallest possible number of rows from the T-side to perform the join.

Semijoin Option Syntax Examples

Query option hints can be written into the SQL of the query to suggest that the TDV query engine use a semijoin in the query. For example:

```
SELECT column1 FROM table1 INNER {OPTION SEMIJOIN} JOIN table2 ON
table1.id = table2.id
```

A query hint, enclosed in curly braces immediately precedes the JOIN keyword.

The default value of OPTION SEMIJOIN is True, so the value of semijoin does not have to be explicitly set. You can also specify something more specific like:

```
… INNER {OPTION SEMIJOIN, PARTITION_SIZE=20} JOIN …
```

This option forces the semijoin to be partitioned, with each partition having no more than 20 elements.

Setting Semijoin Configuration Parameters

These server configuration parameters control what the TDV query engine considers a query that can benefit from semijoin optimization.

The cardinality of both sides of a potential semijoin are evaluated, and the side with smaller estimated cardinality is loaded into memory as the LHS. When the cardinality is small enough, one IN clause or an OR expression is created containing all the values in the join criteria from the LHS, which is then added to the SQL sent to the RHS.

Semijoin is limited for databases whose vendors restrict how large a SQL statement or IN/OR clause can be. If the cardinality exceeds specific data source limitations on the size of the IN clause or the OR expression, the query engine creates an execution plan that attempts a partitioned semijoin. The partition breaks the IN list into chunks of 100 or fewer unique values, and multiple queries are executed against the RHS source. If the cardinality is still too large, the system uses the HASH algorithm.

Another restriction is set to keep the LHS from inordinately burdening the join. You can configure the LHS cardinality to determine the row count that trigger an automatic semijoin.

For example, if Max Source Side Cardinality Estimate is 200 and Ratio is 10, a query where source cardinality is estimated at 50 and target at 600 triggers use of the semijoin automatically; a query with source estimate 100 and target estimate 900 does not.

If the TDV query engine can estimate the source side but not the target side, it assumes that the target-side cardinality is large and sets up a semijoin using the value of Max Source Side Cardinality Estimate.

To set the semijoin parameters using Studio

1. Open the Studio Administration menu Configuration option.
2. Modify the values for the following semijoin server configuration parameters as appropriate.

Parameter	Description
Max Source Side Cardinality Estimate	Put an upper bound on the size of the predicate generated by the source side. If the cardinality estimate is greater than this setting, the TDV query engine does not automatically choose a semijoin.
Min Ratio of Target Cardinality to Source Cardinality	Derive a minimum cardinality of the target to trigger automatic semijoin. For example, if the estimate for the source side is 50 and the ratio is set to 12, the target side must be estimated to be at least 600 rows to trigger use of the semijoin optimization. If the cardinality of the target is not specified, the target cardinality is assumed to be very large.

3. Save your changes.

4. Restart the TDV Server.

Configuring Your Data Source for the Semijoin Optimization

The settings on the Advanced tab for a data source help the query engine determine when to use semijoin optimization.

Consider this example:

```
SELECT * FROM DS1.R INNER {OPTION SEMIJOIN} JOIN DS2.S ON R.r1 =
S.s1 AND R.r2=S.s2.
```

Ideally the TDV query engine generates a predicate in the WHERE clause against DS2 that looks like this:

```
(s1, s2) IN ((2,3),(1,5), …)
```

This is called the row constructor form of IN clause.

If the targeted data source (**DS2** in this example) does not support this type of SQL, the TDV query engine has two options:

- Use the **OR** syntax form and generate a query predicate that looks like the following:

```
(s1=2 and s2=3) or (s1=1 and s2 = 5) or …)
```

- Use multiple scalar **IN** clauses syntax to generate a query predicate that looks like:

```
s1 IN (2,1, …) and s2 IN(3, 5,…)
```

The OR syntax is semantically identical to the row-constructor syntax.

The multiple scalar IN clauses syntax is not as selective, and it can result in more rows being brought into the TDV Server for further local processing. The longer the SQL string, the more likely an execution plan has to run a partitioned semijoin. Run multiple statements to retrieve all the data.

To configure your data source for the semijoin optimization

1. Open the data source.
2. Open the Configuration tab.
3. Select the Advanced tab.

4. Scroll down to set the following parameters.

Field	Description
Max Source Side Cardinality Estimate	Change the number of LHS rows that triggers an automatic semijoin. If the left side cardinality is less than this value, and the product of this value and the Min Ratio of Target Cardinality to Source Cardinality (next row of this table) is less than the estimated RHS cardinality, the TDV query engine attempts to rewrite the query execution plan to use the semijoin optimization.
Min Ratio of Target Cardinality to Source Cardinality	Sets a minimum ratio of RHS to LHS to trigger use of semijoin optimization.

5. Save your settings.

Using the Semijoin Option

To define the semijoin in your SQL

1. Gather cardinality statistics on the tables that are part of the semijoin.

   The statistics are used for the execution plan.

2. Include the LEFT_CARDINALITY query hint and its value.

   The query optimizer uses the hint to choose a better query plan. For example:

```
SELECT column1 FROM table1 INNER {OPTION LEFT_CARDINALITY=10}
JOIN table2 ON table1.id = table2.id
```

   If the query optimizer knows that LHS cardinality is small enough compared to RHS, it attempts a semijoin.

   If LHS cardinality is less than the value of Max Source Side Cardinality Estimate, and the product of LHS cardinality and the value of Min Ratio of Target Cardinality to Source Cardinality is less than the estimated RHS cardinality, the TDV query engine attempts to rewrite the query execution plan to use the semijoin optimization.

3. Define the RIGHT_CARDINALITY query hint and its value.

   The optimizer uses the hint to choose a better query plan. Specifically, it is used when checking LHS cardinality against the minimum ratio set in Min Ratio of Target Cardinality to Source Cardinality.

```
SELECT column1 FROM table1 INNER {OPTION RIGHT_CARDINALITY=10000}
JOIN table2 ON table1.id = table2.id
```

If the RHS cardinality of a table is not known or not specified, the TDV query analyzer assumes that it is large, and determines whether to use semijoin according to the specified cardinality of the LHS.

4. If statistics have not been collected on the data source, optionally use Studio to specify the minimum, maximum, and expected cardinalities.

Semijoin with Non-Equality Conditions Scenarios

Semijoin optimization can also apply to joins that use non-equality conditions (non-equi-joins).

Ideally, full statistics and cardinalities have been gathered on the tables, or predicate cardinality has been specified explicitly by query option in the view SQL. The semijoin can be used when tables are joined on an equality or non-equality condition. Non-equality semijoin optimizations require that table cardinality be specified; otherwise, the query is forced to use a less than optimum merge strategy.

For example:

```
SELECT * from DS1.R LEFT OUTER JOIN DS2.T on r1 > t1 and r1 <= t2
```

The target side predicate always uses the OR form, and looks like this:

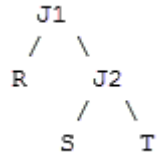
```
(1 > t1 and 1 <= t2) OR (6 > t1 and 6 <= t2) …)
```

The semijoin target is always the immediate child of the join.

In some cases the target is not the immediate child of the join. For example:

```
SELECT * from DS1.R INNER JOIN DS2.S INNER JOIN DS3.T on s1=t1 on
r2=s2
```

The topology for this query example looks like this:



Two semijoins are possible. J2 can have S as source and T as target, and J1 can have R as source and S as target. Both joins can use the semijoin optimization at the same time. The definition of source and target of a semijoin needs to be refined.

For an inner join or a left outer join with semijoin, the immediate LHS child of the join is the source, and the target is a FETCH against a data source found in the RHS sub-tree of the join. The ON clauses that define table connections determine the specific FETCH node target. For right outer join, the definition is the reverse.

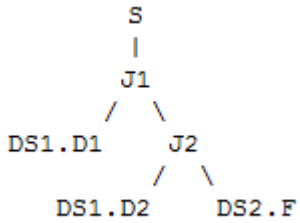
A join with a semijoin optimization can have more than one target. For example:

```
SELECT * from DS1.D1 INNER JOIN DS2.F INNER JOIN DS1.D2 on ds1=f1
on ds2=ds1
```

The join with D1 as source can target both F and D2. Because multiple nodes target DS1, this join can be evaluated as a star schema semijoin.

Star Schema Semijoin

A star schema semijoin is like a query with multiple joins. Consider this query topology.



Here the ON clauses of the joins are such that both D1 and D2 are connected to F, and both joins are inner joins. In this case two semijoins are possible—one from D1 to F and the other from D2 to F—because they both target the same data source.

If the data source is sufficiently robust, the data source can be marked in TDV Server so that the query engine is aware that the data source supports star schema semijoin optimization for multiple join nodes. In Studio, the Info > Advanced tab of the data source configuration pane has a Supports Star Schema property.

When the Supports Star Schema check box is enabled, the TDV query engine is made aware that both joins can be run using the semijoin optimization.

The reason this needs to be explicitly enabled is that even one semijoin with a large source side can place a significant burden on the target data source, and if several joins target the same data source at the same time the burden can overwhelm some data sources. It is easier for star schema semijoin to generate SQL strings that exceed the capabilities of the target data source. Thus this setting is useful when the target data source is very powerful or when all source sides are fairly small.

If the SQL sent to the target side in a star schema query exceeds the maximum length of SQL for the data source, the engine runs with a partitioned semijoin if possible, or it disables semijoin optimization for some of the joins.

The partitioned semijoin might not always apply to a star schema semijoin. If there are *n* joins targeting F and all but one of them generate a short IN clause but one generates a long IN clause, the query engine can still partition the long side of the semijoin. If all the joins produce IN clauses of approximately equal size partitioning might not be useful. In this case, the TDV query engine disables the semijoin optimization for the join that generates the longest SQL string. It continues to disable semijoin optimization until the total SQL for all remaining joins is within the capabilities of the target data source or partitioning becomes an option.

Because of the star schema semijoin, at most one of the predicates can be partitioned. If the query specifies more than one join with the partition_size option, at most only one of these requests can be satisfied.

For information about settings, see Setting Semijoin Configuration Parameters, page 582.

You can override settings for specific data sources.

Data-source overrides apply when a given data source is the target of a semijoin. Typically these values are set conservatively at the server level, and overridden if necessary for specific data sources.

The target data source has the burden of processing the potentially large predicates generated by a semijoin, and the target data source must impose limits on how big the query predicate from the source side can be.

Using Oracle SQL Optimizer Hints

Query performance can sometimes be improved by adding Oracle optimizer hints. If you decide a query hint is necessary, you can use TDV to add the hint to the query syntax; the hint is passed to the database for processing at runtime. The query must be against an Oracle database. The hint syntax must be valid for the version of Oracle that you are using.

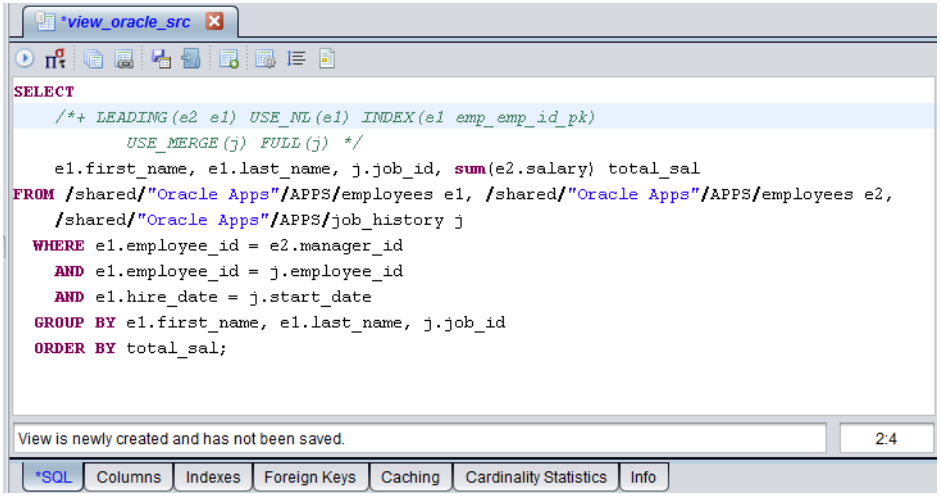
No validation is done by TDV or Oracle on the hint syntax. TDV does not guarantee that the query hint will be pushed, because plan optimization may cause the hint to be dropped. Use the Show Execution Plan button to check whether the hint was pushed or not.

For more information about Oracle optimizer hints, see the *Oracle Database Performance Tuning Guide.*

To add an Oracle optimizer hint to a query retrieving data from an Oracle data source

1. Locate a view in your TDV project that accesses data from an Oracle data source.
2. Select the SQL tab.
3. Enter in the Oracle hint syntax immediately following the SELECT statement.

For example:



4. Save the view.
5. Execute the view to test that the query performs as expected.

Tuning Nested Aggregate Function Behavior

Several data sources support nested aggregate functions, but some do not. Meanwhile, checking for these functions can significantly slow performance. You can use a TDV configuration parameter to decide whether or not to check SQL queries for nested aggregate functions.

For example, Netezza does not support nested aggregate functions, so you can improve performance by setting the Check if nested aggregates is supported by datasource to False for queries pushed to a Netezza data source.

A nested aggregate function looks like this:

```
SELECT AVG(MAX(salary)) FROM employees GROUP BY department_id;
```

To look for nested aggregate functions in SQL

1. Open and log into Studio.
2. From the Administration menu, choose Configuration.
3. Navigate to and select Check if nested aggregates is supported by datasource.
4. Select True.

Tuning Database Channel Queue Size

The DB channel queue size is a setting that specifies the number of buffers within the TDV Server for a client request to prefetch from the database. By default this value is set to 0. If set to 1, the TDV Server can prefetch from the database and put it in a buffer to store it while the TDV JDBC driver is still fetching data from the TDV Server. When the TDV JDBC client asks for the next batch, it is already available in the buffer.

Value	Description of Behavior
1	Data is prefetched and stored in a buffer.
0	Data is only fetched when the client requests it.

To tune database channel queue size

1. Open and log into Studio.
2. From the Administration menu, choose Configuration.
3. Navigate to Server > Client Drivers > Performance.
4. Select DbChannel Queue Size.
5. Specify 0 or 1.

Specifying the Fetch Size for Oracle and MS SQL Server

You can control the amount of data fetched in each batch of rows retrieved from an Oracle or MS SQL Server data source. In the data source capabilities file, the fetch_size attribute controls the number of rows that are being fetched from the data source on every ResultSet.next call.

Adjusting fetch_size is optional and for performance tuning only. Be sure to test the effects of these adjustments to see if performance improves. If the configured value of fetch_size is zero or negative, the default value for the driver is used.

Note: A fetch size that is too small can degrade performance.

To change the batch fetch size for Oracle or MS SQL Server

1. Locate the directory containing data source capabilities files which is:
<TDV_install_dir>\conf\adapters\system\

2. Using a text editor like Notepad, open the capabilities file in the directory for your data source:
microsoft_sql_server_2008_values.xml

oracle_11g_oci_driver_values.xml
oracle_11g_thin_driver_values.xml

3. Uncomment the fetch_size attribute and set its value. For example:
```
<ns5:attribute
xmlns:ns5="http://www.compositesw.com/services/system/util/common"
>
<ns5:name>/runtime/iud/fetchSize</ns5:name>
<ns5:type>INTEGER</ns5:type>
<ns5:value>12345</ns5:value>
<ns5:configID>jdbc.fetch_size</ns5:configID>
</ns5:attribute>
```

4. Save the capabilities file.

TDV Query Engine Optimizations

TDV includes many configuration parameters that can be manipulated to optimize how the server processes the queries that you execute through TDV. With many of these optimizations, TDV does the work to detect how it can best process your SQL. Query engine options let the developer influence the generation of the execution plan by overriding, for specific SQL statements and keywords, TDV configuration settings. The configuration settings can be found in Studio by selecting Administration > Configuration from the main menu and navigating to the parameters under TDV Server > SQL Engine.

Execution of SQL views, procedures, and transactions created with TDV-defined resources follows an optimized execution plan. The execution plan is generated dynamically based on how the SQL is written, what and how native resources are being used, TDV configuration settings, the presence of data-source-specific statistical data, and the presence of TDV SQL query engine options.

Subquery Optimizations

TDV can analyze your subqueries and rewrite them as joins, if they meet the requirements for this subquery optimization. Often better performance can be achieved with joins as opposed to correlated subqueries.

Requirements

- Subquery starts with IN or =.
- Subquery starts with '=' and includes an aggregation.
- The subquery can contain only one selectable of type COLUMN, LITERAL, FUNCTION, or AGGREGATE FUNCTION.
- The subquery can include SELECT, DISTINCT, AGGREGATE, FROM, RELATION and WHERE.
- Co-relation in the subquery can occur only in the WHERE clause for this feature. If co-relation occurs anywhere else, rewrite to join won't happen.

- Subquery operator of type SELECT, DISTINCT, AGGREGATE, FROM and WHERE clause. Or a sub-query contains a WHERE clause with a co-related predicate with a HAVING clause.
- Only co-related predicates is supported for the subquery WHERE clause.
- The co-related columns in the queries must come directly from the parent. Columns from a grandparent level are not supported.
- Type mismatches disqualify the subquery for rewriting.

To enable the subquery optimization

1. In Studio, navigate to Administration > Configuration.
2. Expand Server > SQL Engine > Optimizations.
3. Locate the Rewrite IN Sub-Queries as Joins configuration parameter.
4. Validate or change the value to true.
5. Locate the Use Semi-Join While Rewriting IN Sub-Queries as Joins configuration parameter.
6. Review the value of the configuration parameter and determine if you want to change it.
7. Leave the value or make a change.
8. Select Apply or OK.

   The change in the parameter takes effect immediately. All queries that begin processing after the configuration parameter change are analyzed to see if they can take advantage of the optimization. There is no need to restart the server.

Partition or Join Pruning Optimization

Sometimes, for partitioned data, you want to run a query against a specific partition. This behavior is often referred to as join pruning. TDV has been set to perform this for you automatically. If you want to manipulate or confirm the settings, you can use the TDV configuration parameters that are accessed through Studio.

Requirements

- Null values are not allowed in the tables for join pruning consideration.

To review the join pruning optimization

1. In Studio, navigate to Administration > Configuration.
2. Expand Server > SQL Engine > Optimizations.
3. Locate the Disable Join Pruner configuration parameter.
4. Validate or change the value.
5. Select Apply or OK.

   The change in the parameter takes effect immediately. All queries that begin processing after the configuration parameter change are analyzed to see if they can take advantage of the optimization. There is no need to restart the server.

DATA_SHIP_MODE Values

DATA_SHIP_MODE is a SELECT option that controls automatic rework of federated queries across data sources. Reworked table selections can be shipped through an API to temporary tables so that query nodes can be joined with local tables.

DATA_SHIP_MODE modifies how the query engine handles queries that are candidates for data ship optimization.

For more information, see Defining TDV Data Ship Configuration Parameters, page 604.

Performance Configuration Parameter Tips from an Expert

When using several of the TDV performance enhancement features, such as caching and the data ship options, some of the values for the configuration parameters can create a conflict. In particular, pay attention to:

Configuration Parameter	Description
DBChannel Prefetch Optimization	Improves the performance for read/write operations. By default, this parameter is set to TRUE.
DBChannel Queue Size	Allows you to tune the number of buckets. By default, this parameter is set to 0. The appropriate starting value **is 10**.

If the combination of DBChannel Prefetch Optimization set to TRUE and DBChannel Queue Size set to 1 exists in your environment, new sessions fail to be created after many sessions and requests terminate due to an exception. This is because only 1 prefetch bucket is created. Restarting the TDV may temporarily clear the issue, but it is best to adjust the values of the configuration parameters.

Examine your DBChannel Queue Size setting and increase it as appropriate.

Data Ship Performance Optimization

The data ship optimization analyzes your queries to determine if performance gains can be realized. By temporarily transferring a small amount of data to a data source target, the native data source query optimizations can be used to significantly improve performance.

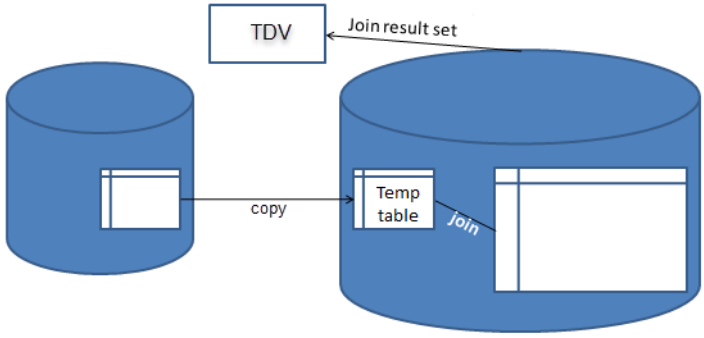
About Data Ship

The data ship optimization accomplishes efficient federated query execution by transformation of federated queries into locally executed SQL operations. When SQL operations involve a very large table (with millions or hundreds of thousands of rows) and a small table with significantly fewer rows, the TDV Server can enhance query performance by shipping a copy of the smaller table to the data source of the larger table. This optimization can yield significantly faster results than a traditional federated query.

The following SQL operators can take advantage of the data ship optimization:

- JOIN
- UNION
- INTERSECT
- EXCEPT

When data ship optimization is enabled the TDV query engine evaluates a query, which can include explicitly defined query option hints, with all relevant data source statistics gathered on query dependencies to determine whether using the data ship optimization yields best performance. If data ship optimization can yield faster execution times according to the analysis and settings, an execution of the query sends data source specific instructions to move copies of tables (or results sets from an execution) from the database with the smaller table to a temporary table on the target data source. Processing of the query occurs on the data source with the larger source table. Because the two sides of the federated query reside on a single data source, processing efficiencies yield a faster return of the result set.



When a query is submitted to TDV, TDV evaluates the query and produces an execution plan without using the data ship optimization. This plan is a pre-data ship plan. TDV goes on to examine the fetch nodes of this plan and decides whether to use the data ship optimization. The data ship optimization is used if:

- The data sources corresponding to fetch node supports data ship.
- Data ship is not disabled for the query using query hints.
- Data source is configured as data ship source or target.

Among all the nodes involved, TDV dedicates one node as the target and all others as sources. TDV creates temp tables in the target data source and moves data from other nodes into this temp table. An alternate query plan is generated that transfers the federated SQL operation of a prior plan to the targeted data source.

Determining the target node is based on the:

- Data source corresponding to the fetch node is configured as source or target.
- Cost of retrieving the data with in the lower and upper bound set for data ship.

Data Ship Support and Requirements

The data ship optimization feature requires that at least one data source participating in a join, union, intersect, or except in an eligible SQL execution be configured to act as a target for the data ship, and that at least one other data source participating in the data ship act as the data ship source. The data ship source and the data ship target can be the same type of database or of different database types. For example, you can configure data ship to have an Oracle data ship source and an Oracle data ship target, or you can configure data ship to have an Oracle data ship source and a Teradata data ship target.

Note: Netezza and DB2 assume that if they are used as a data ship target that they can also be the data ship source, therefore if TDV determines that by reversing the direction of the data ship that better performance can be achieved, it will do so regardless of what you may have defined.

Data ship optimization is supported for following data source types.

Data Source Type	Data Ship Source Support	Data Ship Target Support	Performance Option	Notes
DB2 v10.5	Active	Active	Bulk Load using the LOAD utility	LUW
Greenplum 3.3	Active	Active		
Greenplum 4.1	Active	Active		
Greenplum 4.3	Active	Active		
Microsoft SQL Server 2008	Active	Active	Bulk import/export using BCP	
Microsoft SQL Server 2012	Active	Active	Bulk import/export using BCP	
Microsoft SQL Server 2014	Active	Active	Bulk import/export using BCP	
Microsoft SQL Server 2016	Active	Active	Bulk import/export using BCP	
Netezza 6.0	Active	Active	external tables	
Netezza 7.0	Active	Active	external tables	

Data Source Type	Data Ship Source Support	Data Ship Target Support	Performance Option	Notes
Oracle 11g	Active	Active	Database Links	To use an Oracle data source for data ship, the DBA must install the DBMS_XPLAN package in the database and create an area for temporary tables. For this data source to participate in data ship, it must be specified as a data ship source. Participation as a data ship target is optional. If Oracle is both source and target, DB Link needs to be set up between the Oracle databases.
Oracle 12c	Active	Active	Database Links	
PostgreSQL 9.1	Active	Active	Database Links	
PostgreSQL 9.2.3	Active	Active	Database Links	
Sybase IQ 15	Active	Active	Location: iAnywhere JDBC driver	For a Sybase IQ data source to participate in data ship, the QUERY_PLAN_TEXT_ACCESS database option must be set to ON. For this data source to participate in data ship, it must be specified as a data ship source. Participation as a data ship target is optional.

Data Source Type	Data Ship Source Support	Data Ship Target Support	Performance Option	Notes
Teradata 13.00	Active	Active	FastLoad/ FastExport	For this data source to participate in data ship, it must be specified as a data ship source. Participation as a data ship target is optional. Teradata Fastload mode doesn't work correctly using the 14.10 JDBC driver when Teradata is the Target Data Source. To workaround this issue, use the Teradata JDBC 15 driver.
Teradata 13.10	Active	Active	FastLoad/ FastExport	
Teradata 14.10	Active	Active	FastLoad/ FastExport	
Teradata 15	Active	Active	FastLoad	
Vertica 5.0	Inactive	Inactive		
Vertica 6.1	Active	Active	Bulk load utility Export to another Vertica database	

Data Ship Limitations

Some resource constraints limit the use of data ship optimization. Most limitations are related to data type mismatches. Data type mismatches are handled by a transformation of the data, but certain data types from different sources are incompatible for a selected target. Typically, these data type mismatches are most notable for numeric precision.

The following is a list of cases where use of the Data Ship optimization is not recommended.

Data Source Type	Limitation
Microsoft SQL Server	When using Microsoft SQL Server with the bcp utility, you might get data type mismatch errors if you are using BLOB or CLOB data types.

Data Source Type	Limitation
To Netezza	Netezza cannot be the data ship target for source tables containing data of type BINARY or VARBINARY (for example Oracle).
	Netezza cannot be the data ship target for tables with LONGVARCHAR or VARCHAR columns more than 64 KB long, because creation of the temporary table fails.
	When Netezza is the data ship source, data types of FLOAT or DOUBLE might lose precision because of rounding of values sent to a target of a different type.
Netezza to Sybase IQ	With data ship from Netezza to Sybase IQ, NULL values are replaced with zeroes, which results in different query results than when data ship is disabled.
Sybase IQ or Netezza to Oracle	When Sybase IQ or Netezza data sources use Oracle as a data ship target, trailing spaces sent in the shipped table are trimmed in the result sets with the Oracle table.
Sybase IQ to Netezza	When Sybase IQ data sources use Netezza as the data ship target can cause a data mismatch because the Netezza database appends a padding space character in result set data.
To Oracle	If an Oracle database is a data ship target and the transferred data contains UTF-8-encoded East Asian characters, a column length limitation exception can occur. Oracle databases with a UTF-16 character set does not have this problem.
To Sybase IQ	If you are moving data of type FLOAT to a Sybase IQ database, the scale of the data can be lost because of the way that the Sybase IQ JDBC driver handles the FLOAT data type.
Sybase IQ Type4 Driver	Sybase IQ Type4 driver appears to lose the precision of time stamp columns by promoting or demoting it. To avoid this issue, use the Sybase IQ Type2 driver.

Data Source Type	Limitation
Teradata	Teradata controls the number of concurrent FastLoad and FastExport tasks using the MaxLoadTasks and MaxLoadAWT parameters. Excess FastLoad or FastExport tasks are rejected.
	The maximum number of sessions for each FastLoad or FastExport job is limited to the number of AMPs of the Teradata database. Eight sessions work well for most scenarios.
	Teradata's implementation of UNION does not follow the SQL standard, which can result in a data mismatch when the data is shipped to Teradata.
	A row fetch size bigger than 64 KB causes a Teradata error. Refer to Teradata documentation for the best solution to this problem.
	Teradata FastLoad requires that the target table be empty. If the target Teradata table is not empty, JDBC is used to load data into the table.
To Vertica	For Vertica, the maximum length of a BINARY or VARBINARY column is 65000.

Defining TDV Data Ship Configuration Parameters

Several parameters are available for configuring data ship:

- Maximum Number of Concurrent Data Transfers
- Execution Mode
- Buffer flush threshold

Because every system varies, you might need to test several values for these parameters to determine what works best.

To configure data ship parameters

1. Log into Studio as the admin user.
2. From the Administration menu, choose Configuration.

3. Navigate to and determine the best settings for the following parameters.

Parameter	Description
Execution Mode	Default server handling for data ship queries: • EXECUTE_FULL_SHIP_ONLY—(Default) Generates an error to alert you that the full data ship was not successful. Allows data-ship execution only when the query can be performed without federation after shipment. In other words, after shipment the query must be pass-through. With EXECUTE_FULL_SHIP_ONLY, if parts of the query are federated *after* the data ship query optimization has been applied, the query execution plan Data Ship Notes contain a message like the following: Data Ship Query is not possible because after ship query is still federated and Data Ship Mode is EXECUTE_FULL_SHIP_ONLY. • EXECUTE_PARTIAL_SHIP – This option allows both data ship and federated queries to coexist and proceed to execution even if they cannot be fully resolved into a full data ship. EXECUTE_PARTIAL_SHIP allows a query to proceed without throwing an error, even if a federated query is still required to complete query execution. • EXECUTE_ORIGINAL – If certain nodes cannot be pushed and shipped because some predicates cannot be resolved prior to data pass-through, the original (pre-data ship) query plan is executed. EXECUTE_ORIGINAL causes query execution using the pre-data-ship execution plan whenever a query cannot be completely pushed to the data sources. The Data Ship Notes reveal that fact on execution. When a data ship query is not possible because of dependency on external results or to support something that cannot be pushed to the data source, the original pre-data-ship execution plan is used without shipping results to the targeted table to complete the invocation. • DISABLED — causes the query execution plan to process the request invocations without data ship. DISABLED mode is useful for debugging.

Parameter	Description
Maximum Number of Concurrent Data Transfers	Limits the number of concurrent data transfers (default 100,000), to avoid affecting the performance of other processes. Beyond this limit, new queries requiring data transfers are queued until an existing transfer is completed.

4. Click Apply.
5. (Optional) Navigate to and determine the best settings for the following parameters.

Parameter	Description
Buffer Flush Threshold	Limits the size of the buffer. (Default is 10000.) Certain types of data ship SQL executions buffer large tables before delivery to the data ship target, and the buffer size can exceed available memory. The buffer is flushed when this limit is reached.
DataShip Keep Temp File or Table	The default value is false. When set to true, the server does not delete the temp table or file generated for each data ship request. This option should only be enabled when requested by a support team member. This value is locally defined. It is not altered when restoring a backup and is not replicated in a cluster. This parameter is ignored if you are using data ship and check the 'Use global temporary space for temporary tables' check box for the data ship target.

6. (Netezza only) Navigate to and determine the best setting for the following parameters.

Parameter	Description
Keep nzload Log File	The Netezza driver generates a log file for each nzload operation. If true, all log files are kept. If false (default), no log files are saved to disk. This value is locally defined. It is not altered when restoring a backup and is not replicated in a cluster.
nzload Log Directory	The directory to save the Netezza driver nzload log file for data ship. The default log directory is $TDV/tmp/dataship/netezza. This value is locally defined. It is not altered when restoring a backup and is not replicated in a cluster.

Parameter	Description
Escape Character	The escape character to use while exporting contents to or importing contents from a flat file. It defaults to backslash (\).
Truncate Long Strings during native loading	Boolean. True causes any string value that exceeds its declared CHAR or VARCHAR storage to be truncated to fit. False (default) causes the system to report an error when a string exceeds its declared storage.

7. Click Apply.
8. (Teradata only) Navigate to and determine the best settings for the following parameter:

Parameter	Description
DataShip FastExport/FastLoad Threshold	Limit for the FastLoad and FastExport processes within Teradata. If the row number exceeds this limit, use FastLoad or FastExport. This value is locally defined. It is not altered when restoring a backup and is not replicated in a cluster.

9. Click Apply and then OK.
10. Restart the TDV Server.

Configuring Data Ship

Depending on the type of your TDV data source, one or more of the following configuration tasks might be required:

- Configuring Data Ship for DB2, page 608
- Configuring Data Ship Bulk Loading Option for DB2, page 609
- Configuring Data Ship for Microsoft SQL Server, page 611
- Configuring Data Ship for Netezza, page 613
- Configuring Data Ship for Oracle, page 614
- Configuring Data Ship for Sybase IQ, page 615
- Configuring Data Ship for Sybase IQ Targets with Location, page 618
- Configuring Data Ship for PostgreSQL and Vertica, page 618

Configuring Data Ship for DB2

Requirements

- The DB2 database must have EXPLAIN plan tables created within the DB2 SYSTOOLS schema on the database that will participate in TDV data ship.

Limitations

This feature is not valid for:

- Binary and BLOB data types
- Kerberos implementations

About Data Ship and DB2 Temporary Tables

When a DB2 database is the target for your TDV data ship operation, you can ship the data to temporary tables. The 'Use global temporary space for temporary tables' option creates global temporary tables in DB2 and drops the data and table structure after the data ship query is completed.

The temporary table is materialized under the DB2 session schema. CREATE TABLE permissions are required on the database or schema where the temporary table will be created. The schema name is not used in the DDL to create the temporary table. The table structure is saved in the shared data dictionary, when TDV inserts data, the table is created in the session schema, and data is loaded.

When TDV drops the temporary table, it drops the instance tied to the session and the shared table structure.

To configure the DB2 data sources that might participate in data ship

1. Log into DB2 system and start DB2 command processor db2 and call a system procedure:

```
$ db2
```

2. Connect as a user with privileges to create and modify tables within the DB2 SYSTOOLS schema on the database that will participate in TDV data ship. For example:

```
db2 => CONNECT TO <db-name> USER '<db-user>' USING '<db-pass>'
```

3. Determine if EXPLAIN tables already exist in SYSTOOLS schema, using syntax similar to the following:

```
db2 => SELECT NAME, CREATOR, TYPE
  FROM SYSIBM.SYSTABLES
```

```
  WHERE TYPE = 'T' AND (NAME LIKE 'ADVISE\_%' ESCAPE '\' OR NAME
LIKE 'EXPLAIN\_%' ESCAPE '\')
```

If these tables exist, skip the rest of this section.

4. If the EXPLAIN tables do not exist, create them using syntax similar to the following:

```
db2 => CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C', CAST (NULL AS
VARCHAR(128)),
  CAST (NULL AS VARCHAR(128)))
```

5. Verify that the command shows:

```
Return Status = 0
```

6. Verify that the tables were created by using the following syntax:

```
db2 => list tables for schema systools
```

Configuring Data Ship Bulk Loading Option for DB2

For DB2, the TDV bulk loading option makes use of DB2's LOAD utility. The LOAD utility can quickly load or add data to a table where large amounts of data need to move. LOAD can perform significantly faster than IMPORT because LOAD writes formatted pages directly into the database while IMPORT uses SQL INSERTs. Before attempting to use this method, we recommend that you see if using JDBC batch insert options can give you acceptable performance improvements. If you choose to implement the JDBC functionality, you do not need to configure the DB2 LOAD utility, or change any TDV configuration settings.

The DB2 LOAD utility when working with TDV requires that the DB2 command-line utility be run. How it works with TDV varies by platform:

Platform	DB2 Command-Line Utility Name	Execution Details
Windows	DB2CMD.exe	When it runs, a command window pops up, requires a password, and stays active until the upload completes. While the window is open, the password might be visible.
UNIX	db2	It can be run as a background process.

Requirements

- The data being loaded must be local to the server.
- Requires advanced DB2 database level configuration

Limitations

This feature is not valid for:

- Binary and BLOB data types
- Kerberos implementations

To configure the DB2 LOAD utility to work with TDV for data ship

1. Consult the IBM documentation on configuring and using the DB2 LOAD utility.
2. Install and configure all relevant parts of the DB2 LOAD utility according to IBM's instructions for your platform.

   The client drivers might need to be installed on all the machines that are part of your TDV environment.
3. Verify the full path to the DB2 command-line utility that you want to use to run the DB2 LOAD utility. For example, locate the path to your DB2 SQL command-line client.
4. Open Studio.
5. Select Administration > Configuration.
6. Locate the DB2 Command-Line Utility Path configuration parameter.
7. For Value, type the full directory path to the DB2 LOAD command-line program. For example:

   For example: for Windows, set the value to "C:/Program Files/IBM/SQLLIB/BIN/db2cmd.exe"; for UNIX, set it to /opt/ibm/db2/V10.5/bin/db2. If there are spaces or special characters in the pathname, be sure to enclose the whole string in double quotes.
8. Locate the Debug Output Enabled for Data Sources configuration parameter and set the value to True.
9. Locate the Enable Bulk Data Loading configuration parameter.
10. Set the value to True.
11. Click Apply.
12. Click OK.

13. Restart your TDV Server.

    Some configuration parameter value changes are applied while the TDV Server is running, but many of them require a server restart to work correctly.

14. Open Studio and locate the DB2 data source for which you want to enable the bulk load feature.
15. Open the data source editor.
16. Select the Advanced tab.
17. Scroll down to locate the Database Alias (Used For DB2 Load) field.

    This property is passed on to the DB2 Load to identify the cataloged database.

18. For Value, type the name of the DB2 database. For example: dvbudb21.
19. Save your changes.
20. Refresh your cache.

Temporary files used to support this feature are created in <TDV_install_dir>\...tmp\cacheloading\db2, all files, data, commands and logs are deleted by TDV after the upload process completes.

Configuring Data Ship for Microsoft SQL Server

The bulk import and bulk export functionality available for data ship for Microsoft SQL Server requires the configuration of the Microsoft bcp utility. When the bulk import and bulk export functionality is used instead of the JDBC functionality, performance of operations is greatly improved. The BCP utility (bcp.exe) is a command-line tool that uses the Bulk Copy Program API. The Microsoft bcp utility performs the following tasks:

- Bulk exports data from a SQL Server table into a data file.
- Bulk exports data from a query.
- Bulk imports data from a data file into a SQL Server table.
- Generates format files.

If you choose to implement the JDBC functionality, you do not need to configure the bcp utility, or change any TDV configuration settings.

When configuring data ship or a cache for Microsoft SQL Server, you can configure the bcp utility and then set up access permissions. Both of these processes are described below.

The configuration steps are very similar for caching and data ship, for more information see Configuring Native Caching Option for Vertica, page 519.

Note: Note: In the bcp utility, NULL values are interpreted as EMPTY and EMPTY values are interpreted as NULL. TDV passes the '\0' value (which represents EMPTY) through the bcp utility to insert am EMPTY value into the cache or data ship target. Similarly, TDV passes an EMPTY string ("") to the bcp utility to insert a NULL value into the cache or data ship target.

To configure the Microsoft bcp utility to work with TDV for data ship

1. Verify that bcp.exe has been installed in an accessible directory.

   Note the full path to the bcp.exe file.
2. Open Studio.
3. Select Administration > Configuration.
4. Locate and select the Microsoft BCP utility parameter.
5. For **Value**, type the full directory path to the bcp.exe. For example:

```
C:\Program Files\Microsoft SQL Server\100\Tools\Binn\bcp.exe
```

6. Optionally to use Windows Authentication, locate and select the DataShip BCP Threshold parameter.

   The default value is 50000. After bcp utility is enabled, if the data shipping cost estimate is 50000 or above, data ship uses the bulk import and export implementation; otherwise JDBC, is used to move the data.
7. Optionally to use Windows Authentication, adjust the value of the DataShip BCP Threshold parameter to values that fit your requirements.
8. Locate and select the DataShip BCP Threshold parameter.

   The default value is 50000. After bcp utility is enabled, if the data shipping cost estimate is 50000 or above, data ship uses the bulk import and export implementation; otherwise JDBC, is used to move the data.
9. Adjust the value of the DataShip BCP Threshold parameter to values that fit your requirements.
10. Click Apply.
11. Click OK.
12. Stop the TDV Server.
13. From the command line where your TDV Server runs, log in as the domain user.
14. Start the TDV Server.
15. Click OK.

16. Optionally, when running with Windows Authentication, on windows, open Services > TDV Server <version> and select Properties.
17. Select the Login On tab, set the This account fields to a domain user who has access to MSSQL instance.
18. Click OK.

To set up access permissions for SQL Server for data ship

1. Consult your SQL Server documentation for how to set the SHOWPLAN permissions for the database.
2. For every SQL Server database that will participate in the data ship optimization, grant the SHOWPLAN permissions. For example:

```
GRANT SHOWPLAN
TO <database> [ ,...n ]
```

3. You can now complete the setup in Finishing Data Ship Configuration for Data Sources, page 618.

Configuring Data Ship for Netezza

All Netezza data source connection configurations must be set to take advantage of the data ship optimization, see Netezza Data Source Limitations, page 82. You can now complete the set-up in Finishing Data Ship Configuration for Data Sources, page 618.

Optionally, if you need to move data that has NUL characters, you can use the following procedure to determine how those characters characters are managed by TDV.

When the Ignore Nul Characters in Strings configuration parameter is true, NUL('\0') characters get discarded by Netezza and rest of the String is loaded. When the configuration parameter is false, NUL characters in Strings are not discarded. The default value is false.

To ignore Nul characters in strings

1. Make sure you have both Modify All Config and Access Tools rights.
2. Log into Studio as the admin user.
3. From the Administration menu, choose Configuration.
4. In the tree pane, navigate to Data Sources > Netezza Sources > Ignore Nul Characters in Strings.
5. Select True.

When set to false, NUL characters in strings are not discarded.

6. Click Apply.
7. Click OK.

   This Studio configuration change is not immediately propagated to other open instances of Studio connected with this server.

8. Restart the TDV Server.

Configuring Data Ship for Oracle

Data ship makes use of explain plans. If your Oracle database does not already have plan tables designated to hold that data, then for data ship to work, you must create them.

To configure data ship for Oracle

1. As a user with DBA rights for your Oracle databases involved with data ship, open your favorite tool to use to create tables.
2. Create one or more tables to hold explain plan data.
3. Make sure that the TDV users that are designing your data ship sources and targets has the following privileges:
   - create plan tables
   - execute explain plans

If this error occurs, permissions might still need adjusting:

```
An internal error has occurred.Cause: For input string: ""
```

If data is being moved between Oracle instances, performance is significantly better when database links are used instead of JDBC. Using Oracle database links allows the Oracle instance to take advantage of the proprietary channels for transferring large data tables from one Oracle instance to another.

The following steps are optional, but recommended.

To use database links for the data ship

1. Grant the TDV-Oracle database user privilege to execute the following query:

```
SELECT * FROM dba_db_links;
```

2. Create a database link in the Oracle client using syntax similar to the following on the Oracle instance:

```
CREATE DATABASE LINK oral2_dblink connect to devdata identified by
password using '(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL
```

```
= TCP)(HOST = <host_name>)(PORT = 1521)) ) (CONNECT_DATA = (SERVER
= DEDICATED) (SERVICE_NAME = <service_name>) ))';
```

3. Verify a connection with the new database, oral2_dblink, by executing a query like the following:

```
SELECT * FROM dual@oral2_dblink
```

4. Continue with the instructions in Finishing Data Ship Configuration for Data Sources, page 618.

Configuring Data Ship for Sybase IQ

When Sybase IQ is the target of the data ship, you must perform these steps. Depending on the platform you are installing to, your steps might vary. These steps do not detail every step as might be required by Sybase, refer to your Sybase documentation for further details.

To configure data ship for Sybase

1. Verify or install a SQL Anywhere Database Client. You can obtain a trial version through the Sybase download site.

   If you don't have SQL Anywhere Database Client and don't want to install one on your Unix system, then you can copy the client into <TDV_install_dir>/sqlanywhere<ver>, because the SQL Anywhere database client has the drivers that are needed for TDV.

2. Locate the following files in the directory where the Sybase client is installed.
   - jodbc.jar
   - libdbjodbc<ver>.so for Unix
   - dbjodbc<ver>.dll for Windows
3. Copy the files to the locations described in the following table:

OS	Copy jodbc.jar Into	Copy dbjodbc<ver>.dll Into
Windows 64	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib\win64
Windows 32	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib
UNIX (32 and 64)	<TDV_install_dir>\apps\common\lib	<TDV_install_dir>\apps\common\lib

4. Stop the TDV Server.
5. For UNIX, set the global LD_LIBRARY_PATH environment variable to point at the SQL Anywhere client libraries folder. TDV uses LD_LIBRARY_PATH to find the Sybase Client library. For example, to temporarily set the variable, type:

```
export
LD_LIBRARY_PATH=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/
```

6. Create an ODBC data source following the guidelines in your Sybase documentation. The following instructions highlight some of the necessary steps depending on your platform

Platform	Instructions

UNIX	1. Create an odbc.ini file. We recommend that you create it under <TDV_install_dir>.
	2. Create a Sybase IQ data source name (DSN) in the odbc.ini file. For work with TDV, the Driver variable is the most important setting, because it points to the SQL Anywhere client that you have installed. For example, data sources named, test1 and test2, would look similar to the following:
	`####################################`
	`SybDB@machine64:cat odbc.ini`
	`[test1]`
	`Driver=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/libdbo`
	`dbc<ver>_r.so`
	`host=10.5.3.73`
	`port=2638`
	`uid=dba`
	`PWD=password`
	`DatabaseName=asiqdemo`
	`PreventNotCapable=YES`
	
	`[test2]`
	`Driver=/opt/<TDV_install_dir>/sqlanywhere<ver>/lib<ver>/libdbo`
	`dbc<ver>_r.so`
	`host=10.5.3.74`
	`port=2638`
	`uid=dba`
	`PWD=password`
	`DatabaseName=asiqdemo`
	`PreventNotCapable=YES`
	`####################################`

Platform	Instructions
Windows	1. Start the ODBC Administrator, select Sybase > Data Access > ODBC Data Source Administrator. 2. Click Add on the User DSN tab. 3. Select the Sybase IQ driver and click Finish. 4. The Configuration dialog box appears. 5. Type the Data Source Name in the appropriate text box. Type a Description of the data source in the Description text box if necessary. Do not click OK yet. 6. Click the Login tab. Type the username and password. If the data source is on a remote machine, type a server name and database filename (with the .DB suffix). 7. If the data source is on your local machine, type a start line and database name (without the DB suffix). 8. If the data source is on a remote system, click the Network tab. Click the check box for a protocol and type the options in the text box. 9. Click OK when you have finished defining your data source.

10. On Unix, use the following commands to verify that the DSN is set up successfully:

```
cd sqlanywhere<ver>/bin<ver>
./dbping -m -c "DSN=test1"
```

    A "Ping server successful" message means that the DSN is working. At this point any application can use the DSN to contact the Sybase IQ through the ODBC Driver.

11. On Unix, set the global ODBCINI environment variable to point at the SQL Anywhere odbc.ini file. For example, to temporarily set the variable, type:

```
export ODBCINI=/opt/<TDV_install_dir>/odbc.ini
```

12. Set the QUERY_PLAN_AS_HTML Sybase option to off. For example, run the following SQL command:

```
SET OPTION <USER_NAME>.QUERY_PLAN_AS_HTML = OFF;
```

    This setting prevents Sybase from creating extra HTML plan files that can take up too much disk space.

13. Set the QUERY_NAME option to specify a name for each of the queries generated to help with future database cleanup.

14. Start TDV server.
15. Continue with the instructions in Finishing Data Ship Configuration for Data Sources, page 618.

Configuring Data Ship for Sybase IQ Targets with Location

JDBC can be used to transfer tables, but significantly better performance can be had with use of location. The following steps are optional, but recommended, when moving data between two Sybase IQ data sources.

To configure Sybase IQ to ship tables using location

1. Create a Sybase IQ server:

```
CREATE SERVER "<server-name_database_name>" CLASS 'ASAJDBC' USING
'<host_name>:2638';
```

2. On the Sybase IQ instance create a user sign-in to allow remote connections:

```
CREATE EXTERNLOGIN <user_name> TO "<host_name>_<database_name>"
REMOTE LOGIN <user_name> IDENTIFIED BY <password>;
```

3. Configure the added server info in the interfaces file for the Sybase IQ server.
4. Repeat the configuration for each Sybase IQ server, pointing the SQL location at each of the other Sybase IQ instances.
5. Continue with the instructions in Finishing Data Ship Configuration for Data Sources, page 618.

Configuring Data Ship for PostgreSQL and Vertica

There are no special configuration steps necessary for the data sources outside of Studio.

Finishing Data Ship Configuration for Data Sources

All data sources that are to participate in data ship should be configured as data ship targets. If only one node is configured to be a data ship target, the relative sizes of the nodes are not considered; the SQL operation can only be performed on the data source that can be the data ship target.

At least one data source in a federated join must be configured to accept data shipments of external tables as the target; otherwise, data ship optimization is not possible.

Data can only be shipped in the direction of the data ship target.

When all participating data sources are configured to act as a data ship target, the estimated costs to obtain results from each of the nodes is analyzed. The node that is likely to cost the most to retrieve is designated as the target and nodes with smaller cardinality are fetched to ship to the target node. The SQL operation is performed locally on the temporary tables in the data source to gain the efficiencies available on the target.

About Lower and Upper Bounds

When data-source-specific analytic or aggregate functions are used in queries, set the lower bound to zero so that nodes with very low cardinality results get pushed to the data source for shipment. For joins where the nodes have low cardinality and do not use data-source-specific functions, set the lower bound to block data shipment of results with only a small number of rows. Performance is ordinarily better in this case because the query initialization and temp table creation can take a few seconds longer than direct processing of a small number of rows that do not require SQL analytical processing.

Where more than two nodes are involved in a SQL operation, it might not make sense to limit shipment of a small result set, because external table creation and shipment can happen in parallel. The SQL operation is not hindered by gathering relatively small results tables from multiple sources while a larger data set is loading.

The upper bound value sets the size (as estimated and measured by rows) limiting what result tables can be considered for shipping as external tables.

A value of zero disables data shipment for that data source, but does not disqualify it from participating in a data ship as the target for other results tables. If a query node estimate is greater than the upper bound for that data source, it can only participate as the data ship target. The upper bound is data-source-specific; it sets the bounds to restrict the movement of very large result tables.

To finish data ship configuration

1. Open Studio.
2. Right-click the data source name in the Studio resource tree and select **Open**.

3. Select the Configuration tab, and provide values for the following on the Basic tab:
   - **Pass-through Login**—For non-Netezza data sources, set to **Disabled** so that data ship functionality can work properly.
   - **Transaction Isolation level**—Set to **Serializable** for data ship functionality. This prevents dirty reads, non-repeatable reads, and phantom reads.
4. Use the **Advanced** tab to set values for the following fields. The fields that you see vary depending on the type of data source with which you are working. For example, a Sybase IQ data source might have the Sybase iAnywhere JDBC driver check box displayed, but a Teradata data source would not

Field	Type of Data Source	Description
Is dataship source	• DB2 • Oracle 11g • Sybase IQ 15 • Teradata • MS SQL Server • Vertica	This must be checked if the physical data source might be used as a source of shipped tables to another data ship enabled data source. Check **Is dataship source** for all data sources so that the TDV Server can analyze the query and determine the side that would be best to ship to based on expected or estimated query node cardinality.
Is dataship target	All	This must be checked if the physical data source might be used to receive shipped tables from another data ship enabled data source. Check **Is dataship target** for all data sources so that the TDV Server can analyze the query and determine the side that would be best to ship to based on expected or estimated query node cardinality.
Lower bound for data ship Upper bound for data ship LowerBoundForDataShip UpperBoundForDataShip	All	TDV uses **Explain Plan** to arrive at a numeric estimate of the cost of shipping data from a node to the Data Virtualizer. When the cost of shipping a federated query node falls between the limits of the **Lowerbound** and **Upperbound**, it is considered eligible for shipment so that it can be processed locally.

Field	Type of Data Source	Description
Use global temp space for temp tables	DB2	When this data source is the data ship target, this option creates global temporary tables and drops the data together with the table structure after the data ship query is completed.
Schema path for Temp Tables	All	A relative path to set the location of the temp tables on the data source. It is the name of a schema in the data source. Required for DB2, make sure that this name matches a schema name known to TDV. Case should match exactly.
Temp Table Prefix	All	A character string addition to temporary table names so that they are recognized if they are needed. Optional for DB2.
Enable Bulk Import/Export	MS SQL Server	Check box that enables the bulk movement of data using the bcp utility. You must have finished the steps in Configuring Data Ship for Netezza, page 613.

Database Link List	Oracle with database links	Add your database links separated with semicolons, using the following syntax:
		`[DBLINK NAME]@[DBLINK OWNER DS PATH]`
		For example:
		`oral2_dblink@/users/composite/test/sources/dship/DEV-DSJ-ORA11G-2`

Enable Bulk Export/Load	PostgreSQL Vertica	Takes advantage of PostgreSQL COPY command.
Enable PostgreSql dblink	PostgreSQL	Check to enable database links to improve performance if you plan to use this data source for data caching or data ship optimization. If you check this box, add one or more database links by specifying the database link name and path of the data source for each link

Field	Type of Data Source	Description
Sybase iAnywhere JDBC Driver	Sybase IQ	Select this check box to enable better performance. This option enables the Sybase IQ specific ODBC LOAD TABLE SQL tool to import data into TDV. You must have finished the steps in Configuring Data Ship for Sybase IQ, page 615.
Sql Anywhere Data Source	Sybase IQ	Enter your ODBC DSN name.
Enable Sybase IQ SQL Location	Sybase IQ with Location	Select this check box to enable this option for the data source.
SQL Location Name	Sybase IQ with Location	The **SQL Location** name should take the form: `<server_name>.<database_name>`
Path of data source	Sybase IQ with Location	The TDV full pathname to the other Sybase data source. For example: `/shared/sybase_iq_1`
Add Sql Location	Sybase IQ with Location	If there are more than two Sybase IQ instances you can add multiple locations by using the green plus button.
Enable FastLoad/FastExport for large tables	Teradata	Setting this option indicates that you want to use Teradata's FastLoad or FastExport utility to speed up your query times. For a given query, cardinality information is used to decide whether to use Fastpath or JDBC default loading.
FastExport Session Count	Teradata	The number of FastExport sessions to use for Teradata.
FastLoad Session Count	Teradata	The number of FastLoad sessions to use for Teradata.
Enable Bulk Load	Vertica	Setting this option indicates that you want to use Vertica's Bulk Load utility to speed up your query times. For a given query, cardinality information is used to decide whether to use Bulk Load or JDBC default loading.

Field	Type of Data Source	Description
Enable Export To Another Vertica Database	Vertica	When set, allows data to be exported to another Vertica database. You need to name each database that is available to accept the exported data. TDV uses the CONNECT and EXPORT commands to establish the connections between the data ship source and the data ship target.
Exported Databases	Vertica	Only available if you have selected the **Enable Export To Another Vertica Database** option.
Exported database name	Vertica	Name of the Vertica database to which you want to export data.
Path of data source	Vertica	The TDV full pathname to the other Vertica data source. For example: `/shared/vertica`

5. Save your settings.

Evaluating Queries for Data Ship Using Execution Plans

Studio execution plans can tell you whether a query can make use of data ship or not. When you create views and procedures that involve data from tables that exist on separate sources the execution plan reveals whether the data ship optimization can improve performance of the query.

During the query development phase, verify that the data ship query plan performs better than a standard query plan without the data ship. Sometimes, the time cost of shipping a very large table across the network between two different kinds of data sources can cost almost as much as processing the query normally.

After configuring your data source to use the data ship optimization, you need to evaluate the queries that are run by that data source to determine if they benefit from the data ship. If the queries benefit from the data ship optimization, there is nothing further that you need to do. If the queries do not benefit from the data ship, you need to disable the feature in the hint of the query SQL.

For more information about the execution plan fields and what they represent, see Performance Tuning, page 551.

To use execution plans to evaluate your queries

1. Open Studio.
2. Open the view or procedure that has the query you want to evaluate.
3. Validate that the query is a SQL operation between two or more data sources that can act as the data ship source and at least one data source that can act as the data ship target receiving the shipped tables.
4. Click Show Execution Plan to reveal the Execution Plan tab.
5. Review the query execution report paying specific attention to:

Detail	Description
Estimated Rows Returned	A value estimated from table cardinalities which are not generally used to estimate the cost of shipping SQL requests. For Netezza, determine the cost of a query with an Explain Plan function. For Netezza, the estimated rows returned field will have a value of Unknown.
Data Ship target	Where to send the results of SQL execution. The presence of a data ship target indicates that data ship will be performed.
Data Ship Notes	A report of the estimated number of rows to be shipped, and the amount of time consumed to get the cost estimate. When a condition blocks the use of data ship, this field usually describes the condition, case, or option that ruled out optimization.

6. Execute other queries to show statistics like costs of table creation and other query initialization costs that add to the overall retrieval time.

See Execution Plan Panel, page 664 for information about the buttons and controls on the Execution Plan. See Working with the SQL Execution Plan, page 553 for information about using the execution plan and evaluating the results.

Using Time Series Functions for Vertica Data Ship Targets

There are a few special steps to consider when defining views that include Vertica time series functions. The following steps are guidelines for how you can make use of the following Vertica time series functions within a data ship target view where the SQL will be run entirely on the Vertica database:

- TIMESERIES SELECT clause which names a time slice
- TS_FIRST_VALUE

- TS_LAST_VALUE

Restrictions

- Running the Vertica time series functions in any data source besides Vertica results in processing errors.
- Time functions with time zone and timestamp functions with time zone, do not have micro- and nanosecond precision, nor time zone information. A workaround for that limitation includes using the CAST function to have the time zone value converted to a VARCHAR value.

To use Vertica time series functions

1. Create or identify a view from any data source for which you want to use the time series functions.
2. If non-Vertica data sources are involved, enable data ship with Vertica as the target. For more information, see Finishing Data Ship Configuration for Data Sources, page 618.
3. Enable data ship with Vertica as a source.
4. Manipulate your view code to make use of the Vertica time series function. Refer to *Vertica Programmers Guide* and *Vertica Reference Guide* for details of the time series syntax. The following view SQL is included as a sample of one way to write a data ship query that uses Vertica time series functions:

```
SELECT
      store_key,  cc_class, case when store_key is not null then
store_key end mycase
  FROM
/users/composite/test/sources/vertica50/store/store_dimension
inner join

/users/composite/test/sources/netezza50/TPCDS100_2/TPCDS100/PROD/C
ALL_CENTER
ON cc_call_center_sk = store_key
 timeseries ts as '3 hour'
over (partition by store_key, first_open_date, cc_class
order by cast(cc_rec_start_date as timestamp))
order by case when ts is not null then ts end  , store_key
```

5. Test the view to make sure that all the SQL is pushed and processed on the Vertica database.

Disabling Data Ship for Specific Query SQL

The data ship optimization is configured for each data source. A data source might have hundreds of queries that it runs. Some of those queries can benefit from having the data ship optimization enabled and so you must configure the data ship. However, there might be a few queries that do not benefit from using the data ship optimization. For those queries that do not benefit from the optimization, you must disable the optimization. For more information on data ship options, see the *TDV Reference Guide*.

To disable data ship for specific queries

1. Open Studio.
2. Open the view or procedure for which you want to disable data ship.
3. Select the **SQL** tab.
4. Add the following hint syntax to the SELECT statement:

```
{OPTION DATA_SHIP_MODE="DISABLED"}
```

5. Save the change.

Managing Open Connection Threads

The driver properties are used to specify connection timeout settings as required by the specific driver. By specifying the properties that are used by your specific data source, you can avoid situations where connections are left open indefinitely. This will prevent TDV threads from locking up on requests that need to establish a database connection.

Because each database has different connection properties, you will need to become familiar with them to determine which property to set and what values are appropriate for your environment.

To manage the connection threads

1. Open the data source editor for the data source that has been identified as having connection threads that stay open too long.
2. Open the Advanced tab.
3. Edit the **Connection Properties—JDBC Connection Properties** to define connection timeout intervals that are appropriate for your environment.

Using Data Ship for Prepared Statements

If data ship has been enabled and a query is submitted using a prepared statement where the query is static but input variables can change, TDV evaluates it for data ship. For queries, TDV computes an execution plan. The execution plan considers data ship for each invocation of a prepared statement after the parameter values are known and presented to TDV.

For queries with prepared statements, the value of each parameter for a given execution is determined, the query is evaluated to build an execution plan, and data ship is considered. If TDV determines that data ship can be used to optimize the performance of the query, then TDV will process the query using data ship.

To use data ship for each invocation of a prepared statement

1. Define a table or view with a PreparedStatement.
2. Make sure that both the source and target data sources have data ship enabled.
3. Publish your table or view.
4. Run the queries containing the PreparedStatement.

Push-Based Incremental Caching

Push-based incremental caching can be set up using the native TDV caching mechanism, plus the TDV Cache Management Service (CMS), to configure a Central Event Server. If you have subscribed to notices from objects that store data in an incremental cache, subscribed clients are notified whenever the result sets are updated. After they are notified, clients can choose to update their result sets or refresh the affected views. Each client instance of TDV can be configured to process the messages and automatically update its incremental cache.

The following resources cannot be cached without being wrapped in a view or procedure:

- Procedures with no outputs. There is no data to cache in this case.
- XML files that have been introspected.
- System tables.
- Non-data sources such as folders and definition sets.

The following topics are covered and should be performed in order:

View Requirements and Restrictions for Push-Based Incremental

Caching

When you decide to use a view for push-based incremental caching, you must consider how the view is defined and how the data source is monitored for change capture.

Subscribed views must have explicitly defined or implicitly inferred keys. Composite views (subqueries) are supported if they occur in the FROM clause. Incremental caches do not support INTERVALDAYTOSECOND and INTERVALYEARTOMONTH data types as key values.

The following view operators are supported.

Operator	Supported
Renaming	Yes. Example: C1 AS C2
Projection	Yes, for creating a view from two table columns with scalar function support. Example: SELECT UPPER(C1)
Selection	Yes, with scalar function support. Example: WHERE LOWER(C1) = 'abc'
Inner Join	Yes.
Left Outer Join	Yes, if only the left side is monitored by GoldenGate. Otherwise, no. The restriction applies because processing of the data change for left outer joins could result in inaccurate data.
Right Outer Join	Yes, if only the right operand is monitored by GoldenGate. Otherwise, no. The restriction applies because processing of the data change for right outer joins could result in inaccurate data.

Requirements for Push-Based Incremental Caching

To set up push-based incremental caching, you must meet the following requirements and perform the described steps:

- The monitored data sources and cache database must be Oracle.

- You must install and configure the following products, using your own licensing and customer agreements with the companies that distribute the products:
  - TIBCO Enterprise Message Service (EMS) and JMS
  - Oracle database
  - Oracle GoldenGate change data capture
  - Apache Geronimo

  This section provides some guidelines for configuring these systems to work with the TDV push-based incremental caching feature, but you must follow the requirements and instructions as detailed in the documentation for each product.
- Installed a valid Active Cluster instance.
- Before you configure push-based incremental caching, you must locate the following:
  - cscms.jar, composite_cdc.jar, cscms_call_propagator.jar
  - tibjms.jar, tibjmsadmin.jar
  - axis2-adb-1.3.jar
- You must also complete the steps and address the suggestions in the following sections:
  - Configuring Oracle Database for Push-based Incremental Caching, page 631
  - Installing and Configuring Oracle GoldenGate, page 633
  - Installing the TIBCO JAR File, page 635
  - Configuring JMS Pump for Oracle GoldenGate Using TIBCO EMS, page 635

Configuring Oracle Database for Push-based Incremental Caching

Data sources and derived tables must be prepared for use in automatically updated incrementally maintained caches and view subscriptions. The caching databases used must set permissions so that tables and indexes can be dynamically created for subscription-related tables.

"Change" messaging from the change-data capture (CDC) service to the specified data source topic is required; without it, no view updates are possible. Refer to Installing and Configuring Oracle GoldenGate, page 633 for information on making the CDC service compatible for use with the TDV Software Change Management Service.

Push-based incremental caching requires an Oracle account that grants the following permissions to connect with and use the targeted Oracle database as a subscription store:

- Create subscription-related tables and indexes.
- Introspect required tables. Both **READ** and **SELECT** access are required.
- Execute regular SQL queries against data sources and tables of interest.
- Grant **EXECUTE**, **INSERT**, **UPDATE**, and **DELETE** permissions against those tables.
- Execute **flashback** queries against data sources and tables of interest.

Oracle databases used to store incrementally cached views and views involving joins of monitored tables must be configured with cursor_sharing set to EXACT. Other settings for cursor_sharing result in degradation of performance.

To configure Oracle data sources

1. Log on to the Oracle database machine as SYSDBA.
2. Turn on supplemental logging at the database level using commands like the following:

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;

SQL> ALTER SYSTEM SWITCH LOGFILE;
```

   Or, log in to the database server as the schema owner for the tables that are to be monitored and change the supplemental logging for individual tables.

```
SQL> ALTER TABLE <tableName> ADD SUPPLEMENTAL LOG DATA (ALL)
COLUMNS;
```

3. Make sure that the Oracle databases have archivelog and flashback enabled. To enable flashback, get the system change number (SCN) value using:

```
SELECT dbms_flashback.get_system_change_number FROM dual;
```

Installing and Configuring Oracle GoldenGate

GoldenGate monitors the Oracle tables for changes, and sends change notification messages to the TIBCO EMS queue so that dependent queries can be updated, and subscribed clients can be notified when data table changes might impact their view.

Access to your Oracle GoldenGate server depends on your implementation environment, and it might require use of an SSH utility like PuTTY or WinSCP. The server address, login, password, and port used are all environment- and implementation-dependent. In the example scripts, parameter files, and code given, the GoldenGate installation directory, $GGHOME, is /opt/oracle/ggs/, but that is configurable.

The three *.prm file parameters, GETUPDATEBEFORES, NOCOMPRESSUPDATES, and NOCOMPRESSDELETES, must be included, because these settings control the format of the change notification messages so that they are sent in the format expected by TDV.

If any of the parameters are omitted, then the message content and message format might be malformed, and all incrementally maintained caches might be marked as invalid to prevent use of bad data. Malformed change messaging from the GoldenGate instance can force the Change Management Service on the Central Event Server to shut down, so that the system can be corrected prior to end-use.

Most of the messaging errors result in invalidation of the caches and termination of CMS until the messaging can be corrected. Error messages are recorded in the cs_server.log.

To configure GoldenGate

1. Install GoldenGate co-located with the Oracle data source that is to be monitored. Refer to the GoldenGate documents for a description of the requirements and installation considerations.
2. Copy the composite_cdc.jar file from the TDV installation directory to the directory:

```
<$GoldenGateHome>/GGIEJ_304/javaue/resources/lib
```

   The TDV CDC JAR inserts a header into the GoldenGate messages and includes the SCN value for the transaction classpath of the Java UE.

3. Log in to the Oracle GoldenGate server and navigate to: $GGHOME/dirprm.
4. Using the GGSCI command line utility, create an `EXTRACT` group.

```
GGSCI (comp-name)
1> ADD EXTRACT <ExtractGroupName>, TRANLOG, BEGIN NOW
```

The string <ExtractGroupName> cannot be more than eight characters.

5. Create as many extracts as required for the monitoring of data source tables. Each extract is run as a separate process that can be set to monitor a different set of tables. Groups of extracts can be started and stopped from an .obey file or by a batch or shell script collectively. See the *GoldenGate Administrator Guide* for details.

6. Extract a trail file by entering the following at the GGSCI command prompt:

```
GGSCI (comp-name)
2> ADD EXTTRAIL <C:\sub-dir\…\$GGHOME\dirdat\XX>, EXTRACT
<ExtractGroupName>
```

XX is the two-character name of the exttrail filename group. All trail files have this prefix with a six-digit incrementing numeral, and they are placed in the $GGHOME/dirdat directory. The GGSCI utility creates a new exttrail and an extract parameter file. This file defines the table resources for monitoring and extraction.

7. Edit the <ExtractGroupName>.prm parameter file, using a text editor, with the following command:

```
GGSCI (comp-name)
3> EDIT PARAMS <ExtractGroupName>
```

8. Save the parameter file in the /dirprm subdirectory.

Example Parameter File

In this example:

- EXTGROUP is the <ExtractGroupName>;
- EG is the two character name of the EXTTRAIL file name group
- TRANLOGOPTIONS controls transaction log options such as the Automatic Storage Management User login to set SQL execution options.
- WILDCARDRESOLVE sets the default explicitly. SQL EXEC sets the NLS_DATE_FORMAT handling so that date data type handling is consistent with the TDV system.
- extract EXTGROUP

```
userid UserName, password password
exttrail ./dirdat/EG
TRANLOGOPTIONS ASMUSER sys@VAULTREPO_ASM, ASMPASSWORD password333
WILDCARDRESOLVE DYNAMIC
SQLEXEC "ALTER SESSION SET NLS_DATE_FORMAT='YYYY-MM-DD
HH24:MI:SS'"
-- The following parameters explicitly set the defaults:
EOFDELAYCSECS 1
FLUSHCSECS 1
```

```
--GROUPTRANSOPS 10
REPORTCOUNT EVERY 1 SECONDS, RATE
-- The following three parameters are required to ensure
compatibility with CMS:
GETUPDATEBEFORES
NOCOMPRESSUPDATES
NOCOMPRESSDELETES

-- Monitored tables are listed here. The following tables are used
as examples.
table qacntrial.address;
table qacntrial.bid;
table qacntrial.category;
table qacntrial.creditcard;
table qacntrial.favoritesearch;
table qacntrial.item;
table qacntrial.sale;
table qacntrial.users;
```

Installing the TIBCO JAR File

As another prerequisite to push-based incremental caching, you need to install the TIBCO JAR file.

To install the TIBCO JAR file

1. Copy the tibjms.jar TIBCO file from the TIBCO installation folder to the directory:
`<TDV_install_dir>/apps/server/lib/`

2. Restart the TDV Server.

Configuring JMS Pump for Oracle GoldenGate Using TIBCO EMS

GoldenGate must be enabled with JARs from both TIBCO JMS and a TDV Java User Exit Program. The TDV Java User Exit Program must also be configured and enabled for use with TIBCO EMS.

To configure JMS pump

1. Copy the tibjms.jar TIBCO file from the TIBCO EMS installation to the directory:
`<$GoldenGateHome>/GGIEJ_304/javaue/resources/lib`

2. Copy the composite_cdc.jar file from the TDV cscms.jar installation directory to the directory:
`<$GoldenGateHome>/GGIEJ_304/javaue/resources/lib`

The TDV CDC JAR inserts a header into GoldenGate messages, and includes the SCN value for the transaction classpath of the Java user environment.

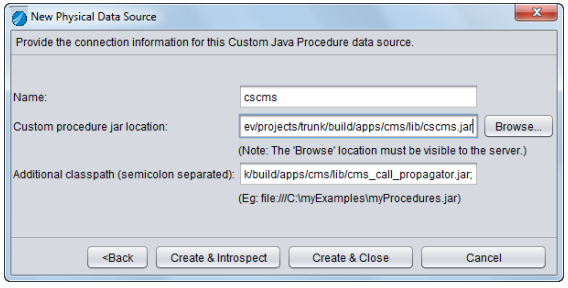
3. Restart the Oracle GoldenGate CDC processes.

Adding the Change Notification Custom Java Procedures

Load the Java procedures to set up change notification and incremental caching. The CMS Change Engine (CE) provides core functionality for the Change Notification feature. For example, it implements and manages the Virtual View Resolver, incremental cache updating, and Change Data Notification (CDN). These features are provided with a TDV Custom Java Procedure (CJP).

To create a new data source based on a custom Java procedure

1. Search for and verify the paths to the following files:
   - axis2-adb-1.3.jar
   - cscms_call_propagator.jar
   - tibjmsadmin.jar
2. In Studio, right-click My Home in the resource tree and select New Data Source.
3. Select Custom Java Procedure and click Next.
4. Enter a name for the new Change Notification Custom Java Procedure and use Browse to specify the location of cscms.jar.



5. In Additional classpath (semicolon separated), type the paths to the JAR files, using forward-slashes ("/") in the path:
   - <TDV_install_dir>/../axis2-adb-1.3.jar;
   - <TDV_install_dir>/../cscms_call_propagator.jar;
   - <TDV_install_dir>/../tibjmsadmin.jar
6. Click Create and Introspect.
7. Select all the procedures listed.
8. Click Next, Finish, and OK.

Defining Connectors

You need to define connectors to establish messaging between the servers, and to ensure fault-tolerance. The connector settings should be identical between the servers in your environment. The Servers require Connector Management configurations after the TDV Server is restarted with the TIBCO interface JAR. The Input EMS and Output EMS could be the same connector on the same TIBCO instance, but because of the amount of traffic expected, and for scalability, input and output messages are best handled by separate queue connectors. The Input EMS is the default Connector handling messages from GoldenGate JMS Pump. The Output EMS handles TDV-to-client change message notifications for subscribing clients.

To define connectors

1. Follow the steps in "Configuring TDV for Using a JMS Broker" in the *TDV Administration Guide* to use Manager to configure TDV for use with a JMS broker.
2. Still in Manager, select CONFIGURATION > Connectors.

   You are going to create a JMS Connector to connect with the TIBCO Queue and a Queue Connection Factory (QCF).
3. Click Add Connector.
4. Type a name for the connector.
5. Select the JMS via JNDI tab.
6. Define the Input EMS JMS connector information.

   This connector takes the messages from GoldenGate and passes them to TDV for processing. Your TIBCO EMS and deployment environment dictates the

values you need to enter for the Connector parameters. Type c in Initial Context Factory and select com.tibco.tibjms.naming.TibjmsInitialContextFactory from the list of factories. Type the JNDI Provider URL (for example, tibjmsnaming://dev-tibco-6:7222).

7. Record the name you gave to the Input EMS Connector.
8. Create another Connector to serve as the Output EMS connector.

   TDV sends messages to subscribing clients through this connector so they can update their caches.
9. Record the name you gave to the Output EMS Connector.
10. Click OK to save the JMS Connector.

Install the Central Server

Push-based incremental caching requires that a central server be defined. In a cluster, one TDV instance is the Central Event Server. Subscribing clients then receive cache increments from the Central Event Server rather than from the Oracle database through GoldenGate.

To install the central server

1. Expand the CMS node you created in the Studio My Home directory.
2. Open the cms_admin_install procedure.
3. Execute the cms_admin_install procedure.
4. Enter CENTRAL as the value of the Mode parameter and click OK.
5. Open Studio > Administration > Configuration and verify that the Change Management Service node (folder) is now present.

Configuring Push-Based Incremental Caching and Change Management

The following Central Event Server settings are required for CMS functionality.

To configure push-based incremental caching and change management

messaging

1. Open the TDV Configuration panel by selecting Administration > Configuration from the Studio menu bar, and navigate to Change Management Service > Central Function > Messaging.
2. Configure Input EMS > Default Connector.

   This is the name of the Input EMS connector you defined using Manager. It is the messaging connection that the Central Event Server uses to receive Change Data Capture data source monitoring change notifications.
3. Configure Internal EMS > Queue Connector.
4. Configure Output EMS > Default Connector.

   This is the name of the Output EMS connector you defined using Manager. It is the messaging connection that the Central Event Server uses to forward change notifications to subscribing clients.
5. Configure Output EMS > Security Domain.
6. Review the configuration parameters under Internal EMS and change them based on the information in Push-Based Incremental Caching Configuration Parameters, page 644.

Publishing the cms_call_propagator Procedure

If you are implementing push-based incremental caching in a cluster environment, you must publish the cms_call_propagator client web services. The Server publishes the procedures cms_client_subscribe, cms_client_unsubscribe, cms_client_renewSubscriptions, and cms_client_getSubscriptions, and the servers use the cms_call_propagator procedure to pass parameters from those calls to the Central Event Server.

For more information on creating a Web service and publishing procedures, see:

- Publishing Resources to a Web Service, page 421
- Publishing Resources to a Database Service, page 416

To publish the cms_call_propagator

1. Create a new service under the Web Services folder; for example, CS_CMS.
2. Locate cms_call_propagator procedure in the Studio navigation tree.
3. Publish it to the Web service you created for it; for example CS_CMS.
4. Click OK.

5. Verify that you can see the procedure in the resource tree.

Configuring Oracle Data Sources for Change Notification

Data sources must have a change notification connector and a topic name specified.

Derived views must be available to be subscribed to, or have cached views that are incrementally maintained. The topic name must be specified on the Change Notification tab of the data source; otherwise, the data source will not receive data source change messages from the Oracle GoldenGate process.

Note: The Change Notification tab for the data source only appears after the Central Event Server has been configured.

The data source Change Notification tab defines the Topic Name to which Oracle GoldenGate is configured to send change notification messages.

Each watched data source must specify a topic on which to receive change notification messaging from the Oracle GoldenGate utility. The topic can use the Input EMS Default Connector or a different JMS Connector specific for that data source.

If a single instance of Oracle GoldenGate has been sending nonconforming messages, it is possible that only the topics affected by that Oracle GoldenGate instance need to be purged rather than all of the data sources topics.

To configure an Oracle data source for change notification

1. Create the Oracle data source.
2. Open the Oracle data source.
3. Select the **Change Notification** tab.
4. (Optional) Type a JMS Connector Name. This value overrides any JMS connection that is defined in the TDV configuration parameters. If no value is specified, the value of the Default Connector TDV configuration parameter is used as the JMS connector.
5. Type a Topic Name. The topic name must be the one you created during JMS configuration. (See Defining Connectors, page 637.)
6. Save the information.

Setting Up an Push-Based Incremental Cache in Studio

For the resource being cached, storage depends on the result set output, and on retrieval of the entire result set or retrieval of a filtered subset. Decide on a storage type for caching the execution result. For information on how TDV cached data types are mapped to data source data types, see “Cache Data Type Mapping” in the *TDV Reference Guide*.

Note: For tables, the expiration period applies to the whole table. For procedures, each input variant’s data has its expiration tracked separately.

If a resource uses a data source that was originally added to TDV Server using the pass-through mode without saving the password, row-based security can affect the cache refresh functionality. For details on Save Password and Pass-through Login, see Adding a Data Source, page 68.

For information about the objects created by the cache, see Caching to a File Target, page 496.

To enable caching, select the cache storage type, and schedule caching

1. Make sure the data source and tables on which the view is based are set up to be monitored for changes and the data source has a connector and topic specified on which change notification messages will be received.
2. In Studio, open a view or procedure.
3. Select the Caching tab.
4. Click Create Cache.
5. Under Status, select the Enable check box.
6. Under Storage, specify the storage type as User Specified.
7. Use Browse to locate and specify the data source. After you select the data source, its full path is displayed in the Data Source field.
8. Use the Open button to open the data source to create two tables, one for storing cache status data and the other for storing cache tracking data.
9. (Optionally) If you want to rename the cache status table, in the Caching section of the data source Configuration tab, select Browse to the right of Status Table.
10. Close the data source Configuration tab. Navigate back to the view or procedure with the open Caching panel.
11. Use Browse to the right of result in the Table for Caching section to specify a location within the data source to create a table for caching.

This location can be the root level of the data source or a schema.

12. In the Select cache table for result window, select the location and type a name under Create Table.
13. Click Create, examine the DDL for result code, and click Execute.
14. Open the table you created to the Caching panel and click Create Cache.
15. On the Caching panel, select Incrementally Maintained.
16. Save the cache settings. After cache settings are saved the resource appears with a lightning-bolt icon in the Studio resource tree to show that the resource is cached.

    After a brief initial data load task, the cache view is actively updated. Status for the cache is **UP** when the resource is saved. If cache initialization fails for any reason, clear the Incrementally Maintained check box, save the view, and then check the box again and save the view again. Because cache initialization is not automatically fault-tolerant, a retry re-initializes the cache.

When a view cache is marked as incrementally maintained and the connection to the source table or the target table caching database instance is down, the time it takes to report the connection problem depends on operating system network settings, such as the TCP default timeout.

If you have data type incompatibilities between your view and your data storage type, see "Cache Data Type Mapping" in the *TDV Reference Guide* to determine the best data storage option for you cache.

Publish Push-Based Incremental Caches

Publishing an automatically updated incrementally maintained cache view is no different from publishing any other view. When an authorized user requests an incrementally maintained view, the result set returned is already synchronized with the data present in the data source. The automatically updated incrementally maintained cache results are served immediately to the user without any additional request or load imposed on the data sources.

Recovering from a Messaging Error

The Change Management Service is stopped when a nonconforming message is received. When CMS is stopped due to a messaging error, all incrementally maintained caches are marked as Disabled, and subscription failure messages with content of "`<ops/>`" are sent to all subscribed clients.

The cs_server.log should be checked for an exception message like the following example:

```
Exception: java.lang.Exception: Invalid message element: expecting
'before' element, got 'missing'...
```

Invalid message element indicates that the messaging format was broken and outside of schema requirements. CMS issues "<ops/>" messages to subscribing clients, invalidates caches, and stops so that CMS and messaging can be restarted from a configured state.

If the exception is a data type mismatch, it is likely that the source of the error was not the Oracle GoldenGate process, and so those processes do not need to be recreated. However, queues and topics might still have to be purged to clear the system of corrupt data.

To resolve messaging issues

1. Check the Oracle GoldenGate configuration parameter file in the /dirprm subdirectory where Oracle GoldenGate is installed. Make sure that the parameters GETUPDATEBEFORES, NOCOMPRESSUPDATES, and NOCOMPRESSDELETES (and any other relevant parameters) are set up to create messages that are compatible with CMS.
2. Stop the "extract" and the "JMS Pump" processes.
3. Delete the GoldenGate extract and JMS Pump processes.
4. Remove the trail files defined in the two related *.prm files. The trail files are typically in the dirdat folder of the GoldenGate home directory, $GGHOME/dirdat, and the files all begin with the two-character prefix specified for the name of the exttrail filename group. (See Installing and Configuring Oracle GoldenGate, page 633.)
5. Make corrections to the configuration parameters.
6. Recreate the GoldenGate extract and JMS Pump processes using the .obey file.
7. If the TDV-Managed Destinations configuration parameter is set to false, use a TIBCO EMS utility to purge queues and topics named under Change Management Service > Central Function > Messaging > Internal EMS, as follows:
   - Use purge queue <queue_name> to purge the queue named in In Transit Event Queue.
   - Use purge topic <topic_name> to purge the topic named in Transient Cache Buffering Topic.

8. Use TIBCO EMS utility purge topic <topic_name> to purge the input EMS topics specified on the Change Notification panels of all data sources configured for incremental caching.
9. Restart the TDV Change Messaging Service on the Central Events Server.
10. Restart all disabled caches with the following for each incrementally maintained cache.
    a. Clear the Incrementally Maintained check box on the Caching tab of the view.
    b. Save the view.
    c. Select the Incrementally Maintained check box and then save the view again. Wait a few moments while the cache re-initializes itself by creating a materialized view based on the SCN timestamp reported by the GoldenGate monitoring utility.

Incrementally maintained caches that were marked as Disabled should then show a status of Up, and those caches will be kept in sync, mirroring all changes made to the watched data sources.

Outbound EMS topics with subscribers listening for change notifications on subscribed views automatically begin receiving change notification messaging when changes to the watched sources are reported.

Push-Based Incremental Caching Configuration Parameters

The following is an alphabetical list and description of the TDV Change Management Service configuration parameters on the Central Event Server. These parameters are visible with an active CSCMS license after CMS has been installed. All of them appear in the Configuration window (select Administration > Configuration from the Studio menu bar) under Change Management Service. In two cases, Logging Options and Tibco, parameters are described in groups.

Parameter/Group	Description
Administration	For a description, see under parameter group Logging Options.
Attempt Count	Under Central Function > Fault Tolerance. The number of times CMS attempts connection recovery after a failure. With the default setting of “0” CMS attempts connection recovery with data sources, EMS, and other active cluster members until all connections are restored.

Parameter/Group	Description
Attempt Delay	Under Central Function > Fault Tolerance. This value denotes the delay, in milliseconds, between attempts to recover from a connection failure. The default value is 1000.
Caching	For a description, see under parameter group Logging Options.
Change Inference	
Change Inference Messages	
TDV-Managed Destinations	Under Central Function > Messaging. Typically, TDV is set to manage output messaging queue and topic destinations. TDV creates queues and topics according to the needs of the client subscriptions, in transit events, and transient cache buffers. If TDV-Managed Destinations is set to false, all of the Messaging > Internal EMS properties must be set.
Connection Attempt Count	For a description, see under parameter group Tibco.
Connection Attempt Delay	
Connection Attempt Timeout	
Data Sources	For a description, see under parameter group Logging Options.
Default Connector	Under Central Function > Messaging > Input EMS. The name of the EMS topic connection factory for receiving change notification messages from Oracle GoldenGate through TIBCO EMS. Data source specific connectors and topics can be defined to override the default connector. The value of the Input EMS Default Connector server property is the name of the JMS via JNDI Connector. JMS connector attributes like initial context factory and JNDI provider URL are defined on the CONNECTOR MANAGEMENT page of the Manager (Web) interface.
Default Connector	Under Central Function > Messaging > Output EMS. The name of the EMS topic connection factory to be used for creating topics to send change notification messages from TDV to subscribed clients. Individual topics are created dynamically based on client subscription requests. Data source specific EMS Connectors and topics can be defined to override the default connector. The value of the Output EMS Default Connector is the name of the JMS via JNDI Connector.

Parameter/Group	Description
Error Handling Procedure	The name of the procedure to call in case an invalid message has been received. The procedure must accept three arguments: a string value for the error, a string value for the offending message, and a timestamp for the time at which the error was reported.
Expiration Period	The number of hours a client subscription remains valid without renewal to monitor a view for changes. Client subscription renewals before the expiration period has elapsed result in extensions that last for another expiration period. The expiration period begins when the client subscription is created and it is reported to the client subscriber.
In Transit Event Queue	Under Central Function > Messaging > Internal EMS. If TDV-Managed Destinations is set to false, this parameter must be set to a valid queue created to host in-transit change events. The In Transit Event Queue is used to ensure fault tolerance so that if a server instance crashes, event processing can resume, starting from the last successfully processed message.

Include Column Names	Under Central Function > Messaging > Message Format. The following example shows an update message ("U") with a format that includes both the rank ("r") and the column names ("n"):
	`<U a="true">`
	`<k r="0" n="Column Name 0">value0</k>`
	`<k r="1" n="Column Name 1">value1</k>`
	`<c r="2" n="Column Name 2">`
	`<n>newvalue2</n>`
	`<o>oldvalue2</o>`
	`</c>`
	`<c r="3" n="Column Name 3">`
	`<n>newvalue3</n>`
	`<o>oldvalue3</o>`
	`</c>`
	`<c r="4" n="Column Name 4">`
	`<n>newvalue4</n>`
	`</c>`
	`<c r="5" n="Column Name 5">`
	`<n>newvalue5</n>`
	`</c>`
	`</U>`

Parameter/Group	Description

Include non-Key Deleted Values	Under Central Function > Messaging > Message Format. Change notification messages always include a primary key that might have one or many columns to ensure a unique key on which create, update, and delete operations were detected. For Delete operations, the message setting can be set to include only key identifying values, or all values, of the row that has been deleted.
	
	If set to true, extraneous column information is included in the change notification message, as in the following example where a delete operation, <D>, includes the key columns (“k”), and non-key values {value3, value4, xsi:nil=”true”}:
	`<ops>`
	`  <D>`
	`  <k r=“0” n=“Column Name 0”>value1</k>`
	`  <k r=“1” n=“Column Name 1”>value2</k>`
	`  <c r=“2” n=“Column Name 2”>value3</c>`
	`  <c r=“3” n=“Column Name 3”>value4</c>`
	`  <c r=“4” n=“Column Name 4” xsi:nil=“true”></c>`
	`  </D>`
	`</ops>`
	
	If set to false, the “c” columns would not appear in the message.

Include non-Updated Values	Under Central Function > Messaging > Message Format. When set to false, change notification messages exclude values that are not changed since the last checkpoint of the table being monitored.
	
	For example:
	`<U a="true">`
	`<k r="0" n="Column Name 0">value0</k>`
	`<k r="1" n="Column Name 1">value1</k>`
	`<c r="2" n="Column Name 2">`
	`<n>newvalue2</n>`
	`<o>oldvalue2</o>`
	`</c>`
	`<c r="3" n="Column Name 3">`
	`<n>newvalue3</n>`
	`<o>oldvalue3</o>`
	`</c>`
	`</U>`

Parameter/Group	Description

Include Old Updated Values	Under Central Function > Messaging > Message Format. If set to true, change notification messages include the old values that are being replaced.
	
	For example, this message highlights the old values, (<o> </o>), that would be omitted if the property were set to false.
	`<U a="true">`
	`<k r="0" n="Column Name 0">value0</k>`
	`<k r="1" n="Column Name 1">value1</k>`
	`<c r="2" n="Column Name 2">`
	`<n>newvalue2</n>`
	`<o>oldvalue2</o>`
	`</c>`
	`<c r="3" n="Column Name 3">`
	`<n>newvalue3</n>`
	`<o>oldvalue3</o>`
	`</c>`
	`</U>`

Logging Options	Parameter group. Typically, logging options are set to false. Log entries and events can be found in the TDV events log (cs_server_events.log) and/or server log (cs_server.log) files present in the server <TDV_install_dir>/logs directory. Each of the following logging options can be turned on (true) or off (false): Administration—CMS installation or uninstallation, CMS start or stop. Caching—Logging for incrementally maintained caching of views. Change Inference—This setting logs messages received through connected EMS topics. The Input EMS Default Connector and change messages are passed from CDC when triggered by changes to watched data source tables. This setting does not show change data in the logs. Change Inference Messages—This setting logs more detailed messages (including data changes) received through connected EMS topics. The messages logged by this setting can help verify whether change data capture messaging is arriving to the EMS topics and connectors currently listening for change messaging. Data Sources—This setting logs activities on the push-based incremental caching data sources. Metadata Inference—Information captured by this log setting shows dependency analysis messages. Subscriptions—Subscription requests, unsubscribe, and renew subscription requests are logged when this setting is true.

Parameter/Group	Description
Message Format	All of the messaging parameters can be turned on (true) or off (false). The parameter settings affect the content of the change notification message format. Message formatting is a global setting that affects all change notification messaging received by subscribed clients.

Message Handling	Correct change notification is dependent on correct change messaging sent from the monitored data source tables. Messaging ensures that incrementally maintained caches and subscribed users have notification of changes in the source data. If messages from the monitored data sources are malformed or lacking in content, the Change Management Service might treat the message as a systemic error requiring manual assessment and reconfiguration, generate logging errors (in cs_server.log), and possibly abort. If messaging errors occur, check the CMS server log file, and check the Oracle GoldenGate *.prm parameter file for configuration.
	`PROCEDURE OnError(IN ex VARCHAR(2147483647), IN msg VARCHAR(2147483647), IN t TIMESTAMP)`
	`BEGIN`
	`  -- Add your code here`
	`CALL /lib/debug/Log(concat('M5 Test[GG erroneous message]---Exception: ',ex));`
	`CALL /lib/debug/Log(concat('M5 Test[GG erroneous message]---Received message: ',msg));`
	`CALL /lib/debug/Log(CAST(concat('M5 Test[GG erroneous message]---Time: ',t) AS VARCHAR(255)));`
	`END`

Metadata Inference	For a description, see under parameter group Logging Options.
On Error Procedure	Sets a procedure to execute when a messaging error occurs. Specify the full path name of the TDV Server procedure to call if an invalid message is encountered. The procedure specified must accept three input arguments: a string value for the error, a string value for the offending message, and a timestamp for the time at which the error is reported.
Persistence Store	The TDV resource path name that points to a relational data source container that should be used to host a subscription store table, which is created and maintained by CMS. On application of a valid TDV persistence store location, CMS creates the cis_cms_appl_subs table. That table records client invocations of cms_client_subscribe with data such as the Subscription ID, Application ID, an encoded expiration, the full path of the subscribed view, and subscribed columns if different from the full view. Because the persistence store, cis_cms_appl_subs table, should not be shared between TDV nodes, the Persistence Store target setting should be changed after synchronization by backup server import.

Parameter/Group	Description
Queue Connector	Under Central Function > Messaging > Internal EMS. The name of the JMS Connector to be used by CMS to create an in-transit event queue. The in-transit event queue is used to maintain in-transit change events in a fault-tolerant manner, while they are being processed by the Central Event server. The value of the Internal EMS Queue Connector server property is the name of the JMS via JNDI Connector. A new or a changed Queue Connector setting requires a CMS restart to initialize the in-transit queue.
Reconnection Attempt Count	For a description, see under parameter group Tibco.
Reconnection Attempt Delay	For a description, see under parameter group Tibco.
Reconnection Attempt Timeout	For a description, see under parameter group Tibco.
Security Domain	Under Central Function > Messaging > Output EMS. The LDAP domain value that can be used to restrict topic connection to permitted groups and users for the subscribed view.
Started	Under Central Function. This setting is an indicator that shows whether CMS is started or not. It should not be modified because the value is not a switch. It is an indicator provided to show CMS status. CMS can be started and stopped with the procedures cms_admin_start and cms_admin_stop.
Stop on Any Error	Under Central Function > Fault Tolerance > Message Handling. CMS aborts on receiving source change messages with an invalid message structure. This flag specifies whether CMS aborts if any other types of errors are detected in a source change message notification. Invalid GG configuration is a systemic error. Partial compliance with the expected change notification message schema is a more serious error than inadvertent messages. Some data type mismatches are handled, but a mismatch invalidates all derived incremental caches and sends a notification to all subscribed clients (<ops/>, set with a message property of aborted="true") to signify a break in the monitored data. When Stop on Any Error is set to false, most messaging errors are still serious, and such errors invalidate all incrementally maintained caches, subscription failure messages, and a purposeful stop of CMS. If a message significantly deviates from the expected CMS schema and is inserted into a monitored input topic or queue, that message is ignored.
Subscriptions	For a description, see under parameter group Logging Options.

Parameter/Group	Description
Tibco	Parameter group. These optional settings override the TIBCO EMS connection factory settings, so that CMS can manage timing and number of connection attempts. The switch from a primary to a backup EMS can initiate a large number of connection and reconnection attempts that can exceed the number allowed. Connection Attempt Count—Number of attempts made to connect to the JMS server. Connection Attempt Delay—Milliseconds to wait before the next connection is made. Connection Attempt Timeout—Milliseconds the connection factory waits before declaring a failure if the attempt is unsuccessful. Reconnection Attempt Count—Number of attempts to make to reconnect to the JMS server. (TIBCO has separate connect and reconnect attempt parameters.) Reconnection Attempt Delay—Milliseconds to wait before the next connection is made. Reconnection Attempt Timeout—Milliseconds the connection factory waits before declaring a failure. For hard connection failures, problems can be reported earlier than these JMS vendor settings indicate.
Topic Connector	Under Central Function > Messaging > Internal EMS. If TDV-Managed Destinations is set to false, this parameter must be set to a valid topic connector for TDV to manage transient cache buffers, which might be needed while an incrementally maintained data cache is being initialized.
Transient Cache Buffering Topic	Under Central Function > Messaging > Internal EMS. If TDV-Managed Destinations is set to false, this parameter must be set to a valid topic so that a transient cache buffer can be set up to provide a temporary topic to handle change messages during cache initialization. The Central Event Server uses this topic to handle change notifications that occur while the cache table is being created and loaded for the first time.

Legacy Web Services

This topic describes managing of legacy web services.The following topics are covered:

Limitations for Legacy Web Service Conversion

If the data service contains a service or operation that TDV does not support, the resource tree does not display that service or operation. For every object that is not supported in TDV, an error is generated in the cs_server.log.

After migration, a report dialog displays messages indicating what needs to be fixed for the web service to be fully functional within TDV. Resolve all warning messages. Consider reconfiguring any older communication protocols between the converted web service and any client applications that consumed the legacy web service.

Limitations Include

- JMS transport is not supported.
- WSDL access authentication method is not supported.
- Combined authentication methods for service endpoint URLs are not supported.
- Multiple services and multiple ports are not supported.

  They are converted to different services, and ports are split into separate web services after migration.
- Wild XML output types are not supported.

  If present they cause conversion of the legacy web service to fail.
- Enclosed keystores are not supported.

  Before upgrade, the enclosed keystore of the legacy web service needs to be integrated with a server key store.

Additional Conversion Tips from an Expert

Legacy web services allowed each Composite web service to have its own keystore to store private and public key pairs.After upgrade, the legacy web service needs to be exported and reimported for the Configuration parameter values that define the key pairs of the TDV global keystore to be configured in the Configuration panel.Because the new service doesn't support separate keystores for each new web service.

Converting Legacy WSDL Data Sources to SOAP Data Sources

If you have existing WSDL data source resources, you can redefine them as SOAP. You must set a TDV configuration parameter. Optionally, after setting the configuration parameter, you can modify your existing data source definitions.

If the WSDL source has multiple services or multiple ports, it is converted to several SOAP data sources based on service and port number. For each SOAP data source, only one service and one port is allowed.

To convert legacy WSDL data sources into SOAP data sources

1. Open and log into Studio.
2. From the **Administration** menu, choose **Configuration**.
3. Search for or select **Import the WSDL data source as a SOAP data source. (Data Sources > SOAP Sources)**
4. Select True to convert all older legacy WSDL data sources into SOAP data sources.

   Click **OK** to save your changes and exit the window.

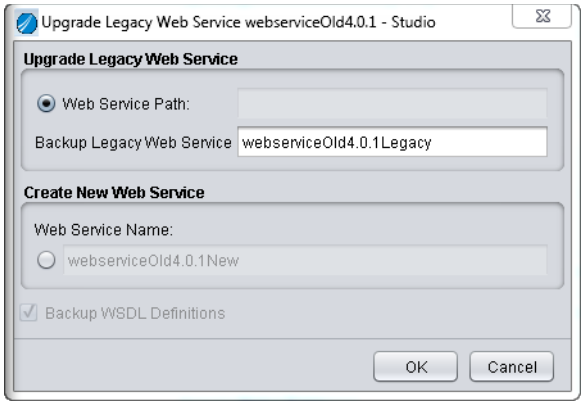
Converting Legacy Web Service

Legacy Web services that you might have can be converted, with some limitations, to the currently used Web services paradigm within Studio. In TDV, a legacy WSDL Web service is an older way to create WSDL that has been retained to support existing implementations.

To convert a legacy web service

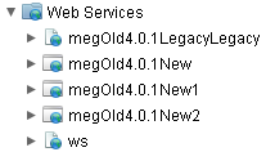
1. Make sure that your web service meets the requirements for conversion.
2. In the Studio resource tree locate your legacy web services.

3. Right click the legacy web service that you want to convert.
4. Select Upgrade Legacy Web Service.



5. Type a name you want to give to the copy of your old web service.

   If warnings are listed, your web service might not convert using the wizard. Consider deleting or rewriting your older web services to comply to the updated TDV web services paradigm.
6. Click **OK** twice to close the window and complete the action.
7. Right click the legacy web service that you want to convert.
8. Select Upgrade Legacy Web Service
9. Select the radio button underneath Web Service Name:.
10. Type a name for the converted web service.
11. Click **OK** twice. Depending on the original structure of your legacy web service you might get more than one new web service. For example:



12. Review and test your new web services as necessary.

Studio User Interface Reference

This topic contains panel, toolbar, menus and other user interface reference material. It includes the following topics:

Studio Menus

The Studio menus contain options applicable to the current window and context. These menus are subject to change and this section might not include listings for options that you see in your version of Studio.

Almost every container or resource in the Modeler resource tree has a right-click menu. The exact listing of options varies by resource type and version of Studio that you are running. See these sections for a quick summary:

File Menu

Available menu options vary depending on where you are in Studio. Many of the options standard to most user interfaces have been omitted.

Menu item	Shortcut
Close Active Editor	Ctrl+W
Close All	Ctrl+Shift+W
Open Lineage for “<resource>”	Ctrl+L
Refresh “<resource>”	F5
Refresh All	Shift+F5
Save All	Ctrl+Shift+S
Save As	Alt+S
Show SQL Scratchpad	Ctrl+Q

Edit Menu

Many of the options standard to most user interfaces have been omitted.

Menu Item	Shortcut
Comment/Uncomment Block	Ctrl+Minus,(Ctrl+Hyphen)
Goto Line	Ctrl+G
Rename <resource>	Shift+F6
Shift-Left	Alt+Left
Shift-Right	Alt+Right

View Menu

Many of the options standard to most user interfaces have been omitted.

Menu item	Shortcut
Last Tab	Ctrl+Alt+Left

Menu item	Shortcut
Next Tab	Ctrl+Alt+Right
Tab Left	Alt+Left
Tab Right	Alt+Right

Resource Menu

Available options vary depending on where you are in Studio.

Menu Item	Shortcut
Find Resource	Ctrl+N
Lock Resource	Ctrl+K
Publish <resource>	Ctrl+P
Unlock Resource	Ctrl+U

Studio Status Bar

The Studio status bar is displayed in the lower right corner of the Studio window. It displays the encryption status of the connection between Studio and TDV. It also displays the current Studio memory usage, and enables you to run JVM garbage collection to free up memory space no longer in use.

Object	Description
Garbage Collection Button	This button initiates a JVM (Java Virtual Machine) garbage collection cycle to free up unused Studio memory. If the TDV Server is on the same computer as the Studio client, this button initiates a garbage collection cycle on the shared JVM.
Studio Memory Use Indicator Bar	This bar shows the total memory available to Studio and the memory it is currently using. Even if TDV Server is on the same computer, this indicator shows only the Studio usage.

Object	Description
Studio-Server Connection Type	A closed-lock icon indicates that SSL protection is being applied to the connection between Studio and the TDV Server.
Encrypted connection— Unencrypted connection—	If the Studio and TDV are on the same computer or if they are both within a secured LAN, it is less important to encrypt the connection between the two. If the connection must be secured, use the Encrypt check box at login to initiate an SSL connection between Studio and TDV.

Studio Code Editing Keyboard Shortcuts

Studio keyboard shortcuts help you write and edit code in the SQL, XML, XSLT, or XQuery editors.

Action	Shortcut	Description
Undo	Ctrl+Z	
Redo	Ctrl+Y	
Comment or uncomment	Ctrl+Minus Ctrl+Hyphen	Adds comment marks to or removes them from selected lines of code. See Commenting SQL, page 218.
Search text in panel	Ctrl+F	Opens a **Search** field at the top of the text editor. All searches wrap at the end to include the entire text. If you type a character that creates a string that is not present in the SQL text, the search field turns red. Up and down arrow-icons let you determine search direction. Check or uncheck the **Case sensitive** box to set case sensitivity.
Search and replace text in panel	Ctrl+R	Opens a window to find and replace a string within the text editor. Options are the same as in the **Search** field and its controls.
Select text to left of cursor	Shift+Home	Selects the text from the cursor position to the beginning of the line.
Select text back to start	Ctrl+Shift+Home	Selects all text from the cursor to the start of the text.

Action	Shortcut	Description
Select text from cursor to end of line	Shift+End	Selects the text from the cursor position to the end of the line.
Select all to end of text	Ctrl+Shift+ End	Selects from the cursor to the end of the text.
Select word left	Ctrl+Shift+ Left_arrow	Selects from the cursor back to the beginning of the word the cursor is in.
Select word right	Ctrl+Shift+ Right_arrow	Selects from the cursor forward to the end of the word the cursor is in.
Select one character	Shift+ Left_arrow Shift+ Right_arrow	Selects one character to the left or right of the cursor or selection.
Move cursor home	Ctrl+Home	Moves cursor to the start of text in the editor.
Move cursor to end of text	Ctrl+End	Moves cursor to the end of the text in the editor.
Shift lines	Alt+ Left_arrow Alt+ Right_arrow	Shifts selected lines left or right one tab setting.
Delete selection	Ctrl+D	Deletes selected text or lines.
Delete one word to left of cursor	Ctrl+ Backspace	Deletes one word (or the remainder of the word) to the left of the cursor.
Delete one word to right of cursor	Ctrl+Delete	Deletes one word (or the remainder of the word) to the right of the cursor.

View and Table Editor Panel Reference

The editor has a collection of right-pane panels for viewing and editing view properties, depending on what is available for the resource you are working with. You can open these panels by clicking tabs at the bottom of the right pane:

- Model Panel, page 662
- Grid Panel, page 662
- SQL Panel, page 662
- Columns Panel, page 663
- Indexes Panel, page 663
- Foreign Keys Panel, page 663
- Cardinality Statistics Panel, page 663

Any combination of the following four panels can appear below the right-pane panels:

- Rebind Panel, page 663, which appears when you click the **Show Rebind Panel** button, so you can bind a view (or procedure) to sources whose names or locations have changed.
- Result Panel, page 664, which appears when results from executing a view (or procedure) execution results are available.
- Execution Plan Panel, page 664, which appears when you click the **Show Execution Plan** button.
- Lineage Panel, which appears when you click the **Show Lineage Panel** button.

After you have created a view or procedure, these four panels are available so you can understand and troubleshoot the view or procedure. You can dismiss them by clicking the button on the right side of the tab.

Model Panel

Use the **Model** panel as a graphical tool to add resources to a view, and to jump-start design of the query that defines the view. For further details, see Designing a View and Table Resource, page 220.

Grid Panel

For details on using the Grid panel, see Designing a View in the Grid Panel, page 230.

SQL Panel

For details on using the **SQL** panel, see Designing SQL for a View in the SQL Panel, page 237.

Columns Panel

See Designing Column Projections Using the Columns Panel, page 245 for more information.

Indexes Panel

For details about using this panel, see Defining Primary Key for a View or Table in the Indexes Panel, page 238.

Foreign Keys Panel

For details about using this panel, see Defining a Foreign Key for a View or Table Resource, page 239.

Cardinality Statistics Panel

The **Cardinality Statistics** panel enables TDV to collect table and column boundary statistics for a cached view (and for data sources and tables) so that the TDV query engine can use those statistics to optimize query execution plans. For details, see Updating the Query Execution Plan, page 562.

The configuration and usage of cardinality statistics is described in Creating Cardinality Statistics for Cost-Based Optimization, page 565. Refer to the topic on Creating Cardinality Statistics for a View, page 566 for information on how this panel is configured and used.

Cardinality statistics are also used to optimize parallel cache loading. See Setting Up the Parallel Cache Option, page 519 for more information.

Rebind Panel

The Rebind panel allows you to rebind the view resources to the appropriate sources if the names or locations of those sources have changed. Display the Rebind panel by clicking the Show Rebind Panel button on the editor toolbar. The Rebind panel displays the schema of the resources that the given view depends on.

For details about using this panel, see Rebinding a View, page 248.

Result Panel

The Result panel appears in the bottom area of a view or procedure editor panel when you click the Execute button on the editor toolbar. For queries, the Result panel displays up to 50 lines of execution results. For some extremely large results, such as those generated by using EXTEND on an array to generate BIGINT number values, some of the data might be truncated without a visual indication by Studio.

For transformations, the Result panel displays the target schema, XML text, or query output.

When you close the Result panel, any query or procedure execution in progress for the Studio display is canceled.

For details, see Generating a Query Execution (Explain) Plan, page 247 and Executing a Procedure in Studio, page 305.

Execution Plan Panel

The Execution Plan panel displays information about the query like an estimate for how much data will be returned, the SQL and projections processed for each command, the data sources involved, and so on. You can also execute the query from the Execution Plan panel. The panel opens when you click the Show Execution Plan button in the toolbar of any view editor panel, the procedure editor panel for a SQL Script or Parameterized Query, or from the SQL tab of the OLAP View editor.

Note: The words “Identity Simulated” appear in the upper right corner of the Execution Plan if row-based security is in use and you are using a simulated identity to test that the results are as expected. This is done using the **Test Identity** tab in the View editor.

See Working with the SQL Execution Plan, page 553 for information about generating and understanding the execution plan.

Procedure Editor Panel Reference

Studio provides a procedure editor for creating and modifying various types of procedures.

This section introduces the following panels:

- Introduction to Procedure Panels and Tabs, page 665
- Model Panel, page 666

The order of the lower-panel tabs (the last four in the list above) depends on the order in which you click a button (in the case of the **Rebind** panel) or execute procedures.

Introduction to Procedure Panels and Tabs

The procedure editor offers a different set of panels for each type of procedure. To open a panel, click the corresponding tab along the bottom of the editor.

Procedure Type	Panels											
	Model	XQuery	Grid	Data Map	XSLT	Inputs	Outputs	SQL Script	SQL	Parameters	Caching	Info
SQL script								x		x	x	x
Custom Java procedure										x	x	x
Packaged query									x	x	x	x
Parameterized query	x		x					x		x	x	x

Procedure Type	Panels											
	Model	XQuery	Grid	Data Map	XSLT	Inputs	Outputs	SQL Script	SQL	Parameters	Caching	Info
Physical stored procedure										x	x	x
XML to tabular mapping										x		x
XSLT transformation				x	x	x	x				x	x
XSLT procedure					x					x	x	x
Streaming transformation				x		x	x				x	x
XQuery transformation	x	x				x	x				x	x
XQuery procedure		x								x	x	x

Model Panel

The **Model** panel is available for Parameterized Queries and XQuery Transformations.

- For a parameterized query, use the Model panel to add resources. You can add any type of table (relational, LDAP, delimited file) and a procedure that outputs either a set of scalars or one cursor. This panel works with the Grid Panel where you can specify the output column projections and the constraints on a query. The Model panel is permanently disabled if you edit the script in the SQL Script Panel for a parameterized query.
- For an XQuery transformation, use this panel for specifying the sources and target values that provide the data for the output XML document. The **Model** panel is permanently disabled if you edit the script in the **XQuery** panel.

For information about using the **Model** panel for XQuery transformations, see Creating an XQuery Transformation, page 320.

Grid Panel

The **Grid** panel in the procedure editor is available for Parameterized Queries. Use this panel for including columns in the output that is obtained when a parameterized query is executed. This panel works with the Model Panel. The Grid panel is permanently disabled if you edit the script in the SQL Script Panel for the parameterized query.

SQL Script Panel

The **SQL Script** panel is available for SQL scripts and Parameterized Queries. This panel has several editing capabilities including keyword high-lighting and syntax formatting. You can also drag resources from the resource tree and drop them into this panel at the cursor position.

Use this panel to formulate and edit the SQL script for a procedure. This panel works with the Parameters Panel. The parameters you define in the **SQL Script** panel must match the parameters designed in the **Parameters** panel, including the order in which they are provided. Editing text in this panel permanently disables the **Model** and **Grid** panels for a parameterized query.

For details about using this panel, see Java Procedures, page 277.

SQL Panel

The **SQL** panel is available for Packaged Queries. Use this panel to formulate and edit the SQL native to the data source targeted for your packaged query.

This panel works with the Parameters Panel. The parameters you use in this panel must match the parameters designed in the Parameters panel, including the order in which they are provided.

For details about using this panel, see Creating a Packaged Query, page 285.

Parameters Panel

The Parameters panel is available for custom Java Procedures, SQL scripts, certain Transformations, Packaged Queries, and Parameterized Queries. This panel displays the input and output parameters for a procedure, and works with the SQL Script panel (or the SQL panel). The parameters you design in this panel must match the parameters defined in the SQL Script panel (or the SQL panel) including the order in which they are provided.

- For Java procedures and transformations, you can only view the parameters in the **Parameters** panel, because the parameters are defined in the source Java code (for a Java procedure) and in the XML or WSDL (for a transformation).

Each parameter is displayed with its TDV JDBC data type and the data type native to the corresponding data source. The output parameters shown in this panel are rendered as columns in the result set when you execute the procedure.

- For a SQL script, packaged query, or parameterized query, use the **Parameters** panel to design and edit the parameters.

Data Map Panel

The Data Map panel is available for certain types of transformations (XSLT and streaming), and lets you transform XML data into tabular data by mapping XML sources to target columns. It displays the hierarchical relationship among the elements in the input XML document with the corresponding data type for each element. The Source column displays the structure of the input XML document, and the Target column the structure of the output table. The output items you specify in the Target column are rendered as columns in the result set when you execute the transformation.

This panel is permanently disabled if you edit the XSLT in the XSLT Panel for XSLT Transformations.

XSLT Panel for XSLT Transformations

An **XSLT** panel is available for XSLT transformations. Use this panel to view the XSLT for the output. The system auto-generates the XSLT when you use the Data Map Panel to design the output for the transformation.

You can edit the XSLT in this panel, but after you start editing the XSLT, the mapping in the **Data Map** panel is permanently disabled. When editing XSLT directly, you can use the same keyboard shortcuts as those used to edit SQL. See Studio Code Editing Keyboard Shortcuts, page 660.

XSLT Panel for XSLT Procedures

An **XSLT** panel is available for XSLT procedures. Use this panel to view the XSLT for the output.

You can edit the XSLT in this panel, or prepare XSLT in any editor and paste it into this panel. When editing XSLT in this panel, you can use the same keyboard shortcuts as those used to edit SQL. See Studio Code Editing Keyboard Shortcuts, page 660.

XQuery Panel

The **XQuery** panel is available for XQuery transformations and procedures. Use this panel to view an XQuery that returns an XML document when the transformation is executed. The system auto-generates the XQuery when you use the Model Panel to design the output for the transformation.

You can edit the XQuery in this panel, but after you start editing the XQuery, the **Model** panel is permanently disabled. When editing XQuery directly, you can use the same keyboard shortcuts as those used to edit SQL. See Studio Code Editing Keyboard Shortcuts, page 660.

Inputs Panel

The **Inputs** panel is available for certain types of transformations (XSLT, streaming, and XQuery). Use this panel to:

- View the input columns if you have used the **Data Map** panel to generate the XSLT for an XSLT transformation.
- Supply global input parameters for an XQuery transformation.

Outputs Panel

The **Outputs** panel is available for certain types of transformations (XSLT, streaming, and XQuery). Use this panel to:

- View the output columns if you have used the **Data Map** panel to generate the XSLT for an XSLT transformation. If you edit the XSLT, disabling the **Data Map** panel, you must use the **Outputs** panel to manually design the output columns for the transformation.
- Supply global output parameters for an XQuery transformation. If you edit the XQuery, disabling the **Model** panel, you must choose an appropriate schema for the output XML document.

The output parameters shown in the **Outputs** panel are rendered as columns in the result set when you execute the transformation.

For details about using the **Outputs** panel, see Adding Target Columns through the Outputs Panel (XSLT), page 319.

XSLT Debug Input Panel

The **XSLT Debug Input** panel is available for XSLT transformations, and opens below the main procedure editor panel when a transformation is executed. Use this panel to view the input to the XSLT. The result columns are displayed with data.

XSLT Debug Output Panel

The **XSLT Debug Output** panel is available for XSLT transformations, and opens below the main procedure editor panel when a transformation is executed. Use this panel to view the output from the XSLT. The result columns are displayed with the created XML.

Trigger Editor Panel Reference

Studio provides a trigger editor for configuration and modification of triggers. You can create a trigger as you would create any other resource: with a right click from an appropriate container within **My Home** or **Shared** folders. The trigger editor opens when you create a trigger. At any time later, you can open a trigger editor by double-clicking the trigger name in the resource tree.

Triggers are also present in a simplified form as part of the configuration panels used for scheduling cache refreshes and cardinality statistics gathering. If more scheduling complexity is wanted for cache refreshes or statistics gathering, then triggers can be created to invoke either the /lib/resource/RefreshResourceCache or the /lib/resource/RefreshResourceStatistics procedure.

When scheduled resources from previous TDV releases are imported, those schedules are displayed as triggers in the Studio resource tree.

The trigger editor has two tabs:

- Condition Tab, page 670
- The Info tab, described in Getting Information about a Resource, page 42

Condition Tab

The Enable Trigger check box is at the top of the Condition tab. This check box must be selected to activate a trigger.

The Trigger Condition tab has four panes in which you specify the trigger information:

- Condition Types for a Trigger, page 392 (upper-left pane)
- Condition Pane, page 671 (upper-right pane)
- Action Types for a Trigger, page 395 (lower-left pane)
- Action Pane, page 672 (lower-right pane)

The selections in the panes on the left (Condition Type and Action Type) determine what options are displayed in the panes on the right (Condition and Action). For example, the trigger editor below shows the options for a condition type of Timer Event and an action type of Send E-mail.

Condition Pane

The contents of the **Condition** pane depend on the condition type selected.

JMS Event Conditions

Prior to use of the JMS event trigger, you need to configure a JMS connector with a queue or topic connection factory for use with TDV. See "Configuring TDV Data Connections" in the *TDV Administration Guide* for information about the JMS configuration requirements. See Creating a JMS Event Trigger, page 396 for how to define a JMS event trigger.

System Event Conditions

You specify the system event name. See Creating a System Event Trigger, page 397.

Timer Event Conditions

You specify the frequency for when the trigger occurs. See Creating a Timer Event Trigger, page 399 for more details.

Only once per cluster—When checked (the default), specifies that the trigger action be performed only once in the cluster. That is, if the trigger action is invoked on one node of the cluster, the other nodes are notified and the trigger action is not invoked again. When unchecked, the trigger action is invoked on every node of the cluster.

User-Defined Event Conditions

You specify the user-defined event name. See Creating a User-Defined Event Trigger, page 402 for more details.

Action Pane

The contents of the **Action** pane depend on the action type (described in Action Types for a Trigger, page 395). A description of the options for each action type are as follows.

Execute Procedure

This action lets you trigger execution of a procedure without sending any notification. The Action pane lists the fields to fill in:

Procedure Path—Specify the procedure to execute:

- Type the procedure path, or open the procedure and copy the path and name from the Name field at the top of the procedure's Info tab.
- Click the Browse button to open a dialog box that lists the resource tree folders that contain procedures. When you make a valid selection, the OK button is ungrayed.

Parameter Values—Specify any parameters and values that the procedure or view uses:

- Click the Edit button to open a dialog box if the procedure takes parameters. Type values in the fields. For fields left blank, check or leave checked the Null box to the right of the field. When you click OK, an ordered, comma-separated list of parameter values is posted to the Parameter Values field.
- Type (or edit after copy-pasting a starter set from the Parameter Values field) an ordered, comma-separated list of input parameter values.

Note: Only static input parameters can be used by the trigger, but the procedure invoked can use whatever variables it might require.

Exhaust output cursors—If the procedure is a SQL script that contains a PIPE cursor, and if the pipe needs to fully execute and load all rows to complete execution of a trigger (for example, to email a result set) checking this box forces the trigger to wait until all the rows are buffered and read before completing. If you leave this unchecked, the trigger returns while the pipe is buffering, cutting off the buffering of the pipe.

Note: As soon as you begin typing text in any field of the dialog box, the NULL box for that field is unchecked. If you decide to delete all text from the field, be sure to recheck its NULL box.

Gather Statistics

This action lets you specify a data source for which to gather statistics data.

Data Source Path—Specify the data source path and name:

- Type the data source path; or open the data source, copy the contents from the Name field at the top of the Info tab, and paste it here.
- Click the Browse button to open a dialog box that lists the resource tree folders that contain data sources. When you make a valid selection, the OK button is ungrayed.

To see how this action works, select the data source to be scanned for data statistics. After creating the trigger, view the **Triggers** console in **Studio Manager** to check if the trigger fired as scheduled.

Reintrospect Data Source

This action lets you specify a data source to reintrospect.

Data Source Path—Specify the data source path and name:

- Type the data source path; or open the data source, copy the contents from the Name field at the top of the Info tab, and paste it here.
- Click the Browse button to open a dialog box that lists the resource tree folders that contains data sources. When you make a valid selection, the OK button is ungrayed.

To, Cc, Bcc, Reply-To—These work the same as in ordinary email applications. You can specify a list of email addresses by separating them with commas (,) or semicolons (;). The email recipients are sent a notification with the result of the output of the procedure or view.

Message Subject, Message Body—These work the same as in ordinary email applications.

Do Not Persist Detected Changes—Check this if you want only a report about the changes detected in the data source, and not save the changes. This option allows the TDV developer or administrator to assess the impact of data source changes to the TDV model before actually applying them.

Do Not Send If No Changes Detected—Check this if you do not want email sent if no changes were detected.

To see how this action works, select the data source to be reintrospected. After creating the trigger, view the **Triggers** console in **Studio Manager** to check if the trigger fired as scheduled. Also verify if the notification was sent to the e-mail recipients.

Send E-mail

This action lets you specify a procedure or view to execute and then send the results through email to one or more recipients. The Action pane lists the fields to fill in:

Resource Path—Specify a procedure or view to execute:

- Type the resource path; or open the resource, copy the contents from the Name field at the top of the Info tab, and paste it here.
- Click the Browse button to open a dialog box that lists the resource tree folders that contain procedures or views. When you make a valid selection, the OK button is ungrayed.

  Note: Before scheduling execution, set the From Address and SMTP configuration parameters.

Parameter Values—Specify any parameters and values that the procedure or view uses:

- Click the Edit button to open a dialog box if the procedure or view takes parameters. Type values in the fields. For fields left blank, check or leave checked the Null box to the right of the field. When you click OK, an ordered, comma-separated list of parameter values is posted to the Parameter Values field.
- Type (or edit after copy-pasting a starter set from the Parameter Values field) an ordered, comma-separated list of input parameter values.

  Note: For an example of a procedure and its parameter values, see Execute Procedure, page 672.

To, Cc, Bcc, Reply-To—These work the same as in ordinary email applications. You can specify a list of email addresses by separating them with commas (,) or semicolons (;). The email recipients are sent a notification with the result of the output of the procedure or view.

Message Subject, Message Body—These work the same as in ordinary email applications.

Include Summary—Check this if you want to include a summary of the notification.

Do Not Send If No Results—Check this if you do not want to receive notification if there are no results.

To see how the **Send E-mail** action works, select the transformation to be executed. After creating the trigger, view the **Triggers** console in **Studio Manager** to check if the trigger was fired as scheduled. Also verify if the notification was sent to the mail recipients.

SQL Definition Set Editor

The SQL definition set editor lets you create your own SQL type definitions, exceptions, and constants.

- SQL Definition Set Editor Toolbars, page 675

See Creating a SQL Definition Set, page 379 for more information.

SQL Definition Set Editor Toolbars

The following table lists the buttons that appear on the **Types**, **Exceptions** and **Constants** panel toolbars for SQL definition sets, and explains what they do. To find out the purpose of any panel button, let the cursor hover over it so its name appears. You can then look it up in the table below, which lists the buttons in their order on the toolbar.

Label	Functionality
Add	Opens a window where you can choose a data type for a type definition you want to add to a definition set. The new type definition is automatically assigned a unique, descriptive name that you can change by selecting Rename from the context menu in the Name column. Available data types are listed in About Definition Sets, page 377.
Change Type	(**Types** panel only) Opens a **Choose Data Type** dialog box in which you can change the column type. Available data types are listed in About Definition Sets, page 377.
Move Parameter Up Move Parameter Down Move Parameter Out Move Parameter In	Moves a column up, down, to the left, or to the right, respectively. These buttons are enabled only if an entry in the definition set is complex such as a CURSOR and it has non-complex base types (for example, binary, decimal, integer, string, time, or array) under it which can be reordered.

XML Definition Set Editor

The XML definition set editor lets you create XML schema definitions. You can import definitions from existing XML files, validate definitions, format definitions, and create definitions from an XML instance.

This editor has two panels—**XML Schema** and **Info**—which are described in the following sections:

- XML Schema Panel (XML Definition Set), page 676
- XML Schema Panel Toolbar, page 676

See Creating an XML Definition Set, page 382 for more information.

XML Schema Panel (XML Definition Set)

The **XML Schema** panel displays the XML schema, element declarations, and type definitions for the selected XML definition set. You can type the schema definitions or import definitions from an existing file, validate the schema, format the schema, or create definitions from an XML instance. You can also add definition source files that can be referenced from other sources in the definition set.

Studio displays keywords in orange type and literal definitions in blue type.

For a table of toolbar buttons, see XML Schema Panel Toolbar, page 676.

XML Schema Panel Toolbar

The following table lists the buttons that appear on the **XML Schema** panel toolbar, and explains what they do. To find out the purpose of any panel button, let the cursor hover over it so its name appears. You can then look it up in the table below, which lists the buttons in their order on the toolbar.

Label	Functionality
Import XML Schema Definitions from File(s)	Opens a dialog box in which to type or browse to the location of a file containing an XML schema definition.
Validate XML Schema Definitions	Requests validation from the definition set editor. If it is not valid, an error box describes the error and its line and column. (The current cursor location is shown in the lower right corner of the panel in line:column format.)
Format XML Schema Definitions	Formats the definition set with hierarchical indentation, and with keywords in orange type and literal definitions in blue type.
Create XML Schema Definitions from XML Instance	Opens a window in which to navigate to an existing XML instance. When you click Open, the XML file appears on the XML Schema panel of the editor.

Label	Functionality
Add Definition Source	Opens an Add Definition Set Source window where you can type a name for a definition source. The name is added to the Select Definition Source drop-down list, from which it can be referenced from other sources within the definition set.
Delete Definition Source	Deletes the definition source currently displayed in the Select Definition Source drop-down list (except XML DefinitionSet).
Rename Definition Source	Opens a dialog box in which to enter a new name for the definition source named in the Select Definition Source drop-down list. All references to the old name will be updated.
Select Definition Source	Provides a drop-down list of available definition sources (XML DefinitionSet plus any you have added).

WSDL Definition Set Editor

The WSDL definition set editor lets you create your own WSDL definition set. You can import WSDL and XML schema definitions, validate definitions, and format definitions. This editor has two panels—**WSDL** and **Info**—which are described in the following sections:

- WSDL Panel, page 677
- WSDL Panel Toolbar, page 678

See Creating a WSDL Definition Set, page 384 for more information.

WSDL Panel

The **WSDL** panel displays the XML schema defined for the WSDL, including type definitions, message definitions, port type definitions, binding definitions, and service definitions.

Studio displays keywords in orange type and literal definitions in blue type.

In the **WSDL** panel, you can type the definitions or import WSDL and XML schema definitions from a file. You can also add definition source files that can be referenced from other sources in the definition set.

WSDL Panel Toolbar

The following table lists the buttons that appear on the **WSDL** panel toolbar, and explains what they do. To find out the purpose of any panel button, let the cursor hover over it so its name appears. You can then look it up in the table below, which lists the buttons in their order on the toolbar.

Label	Functionality
Import WSDL or XML Schema Definitions from File(s)	Opens a dialog box in which to type or browse to the location of a file containing an XML schema definition.
Validate WSDL or XML Schema Definitions	Requests that the definition set editor validate the definition set. If it is valid, a dialog box says so; if it is not valid, an error dialog box describes the error and at which line and column of the definition the error was found. (The current cursor location is shown in the lower right corner of the panel in line:column format (for example **1:100** for line 1, column 100).
Format WSDL or XML Schema Definitions	Formats the definition set with hierarchical indentation, and with keywords in orange type and literal definitions in blue type.
Add Definition Source	Opens an Add Definition Set Source window where you can type a name for a definition source. The name is added to the Select Definition Source drop-down list, from which it can be referenced from other sources within the definition set.
Delete Definition Source	Deletes the definition source currently displayed in the Select Definition Source drop-down list (except WSDL DefinitionSet).
Rename Definition Source	Opens a dialog box in which to enter a new name for the definition source named in the Select Definition Source drop-down list. All references to the old name will be updated.
Select Definition Source	Provides a drop-down list of available definition sources (WSDL DefinitionSet plus any you have added).