IBM Cognos Virtual View Manager
Version 10.2.0

User Guide

IBM
Contents

Introduction.................................. ix

Chapter 1. IBM Cognos Virtual View Manager........................................... 1
Modeling and publishing.............................................................................. 1
Resource management.................................................................................. 1
IBM Cognos Virtual View Manager resource tree........................................ 2
  Resource tree hierarchy............................................................................. 2
  Resource tree contents............................................................................. 2
  Relationship between resources in the resource tree................................. 6
IBM Cognos Virtual View Manager menus.................................................. 7
  File menu.................................................................................................. 7
  Edit menu.................................................................................................. 7
  Resource menu.......................................................................................... 7
  Administration menu.................................................................................. 8
Toolbar.......................................................................................................... 8
Editors and input dialogs.............................................................................. 9

Chapter 2. Resource management basics.................................................... 11
About resource management........................................................................ 11
Resource creation........................................................................................ 11
  Where can you create a resource?............................................................. 11
How to open resources................................................................................ 12
Resource locks and editing resources.......................................................... 13
  New resources in locked containers........................................................ 13
  About locked resources.......................................................................... 13
  Locking change history.......................................................................... 14
  Access to locked resources.................................................................... 14
How to unlock resources............................................................................. 14
How to edit resources.................................................................................. 15
  How to save resources........................................................................... 15
  Moving a resource................................................................................. 16
  Copying a resource................................................................................ 16
  Deleting a resource................................................................................. 17
  Renaming a resource.............................................................................. 17
Resource export and import......................................................................... 17
  Locked resource export and import........................................................ 18
  Resource export....................................................................................... 19
  Marking of rebindables during export...................................................... 20
  Resource import....................................................................................... 21
Using IBM Cognos Virtual View Manager for a full server backup................... 24
Searching for resources in the server............................................................ 24
Viewing table schema and contents............................................................... 25
Resource publishing...................................................................................... 25

Chapter 3. Data sources.................................................................................. 27
Supported data source types........................................................................ 27
Auto-discovering data sources..................................................................... 27
What it means to add data sources............................................................... 27
  Data source wizard................................................................................ 28
  Introspection and re-introspection............................................................ 28
Relational data sources................................................................................. 29
  Procedures for adding relational data sources.......................................... 30
  Adding an IBM DB2 data source............................................................... 30
  Process to add an IBM Cognos Virtual View Manager Data Source........... 35
  Adding DataDirect Mainframe data sources............................................ 35
© Copyright IBM Corp. 2008, 2012
### Chapter 11. Caching resources

- View caching .................................................. 193
- Procedure caching ............................................. 194
- Cache storage type ........................................... 194
- Setting up caches .............................................. 195
  - Cache indexing ............................................... 197
  - Cache configuring and enabling ............................. 198
- Behavior in Virtual View Manager .............................. 198
- Automatic and data source caching ............................. 199
  - Automatic caching ........................................... 199
  - Data source caching ......................................... 199
- Cache data management ........................................ 201
  - Terminology .................................................. 202
  - Scheduled refresh of cached data ............................ 202
  - Expiration of cached data ................................... 202
  - Rules for clearing caches .................................... 203
- Triggers .......................................................... 203
  - Triggers that operate on caches ............................. 203
  - Triggers that detect cache changes .......................... 203
- Transaction result caching ..................................... 203
- Cache lifecycle ................................................ 204
- Enabling and disabling ........................................ 204
- Loading and refreshing ........................................ 204
- Cache clearing ................................................ 205
- Cache status ................................................... 206
- Cache events ................................................... 206
- Configuration changes ........................................... 206
- APIs ............................................................. 207
  - Procedures .................................................. 207
  - Web services ................................................ 207
- Storage tables .................................................. 208
  - The data source status table ................................. 208
  - The data source tracking table ............................... 209
  - Reference columns .......................................... 209
  - Tracking columns ............................................ 209
- Cache data tables ............................................. 209
  - Transaction isolation ........................................ 210
- Best practices and use cases .................................. 210
  - File versus database caching ................................. 210
  - Cache indexing .............................................. 210
  - How to avoid stale data ..................................... 210
- Incremental caching ........................................... 212
  - How to determine what changed ............................... 212
  - Option #1: Transactional database ............................ 212
  - Option #2: Non-transactional updates ........................ 213
- Database specifics ............................................. 213

### Chapter 12. Client interfaces

- Client connectivity information ................................ 223
- Connections via JDBC .......................................... 223
  - Setting the CLASSPATH for csjdbc.jar ....................... 223
  - Properties for the driver connection URL ................. 224
  - Multiple credentials for JDBC connection ............... 225
  - Code sample template ...................................... 226
  - Supported JDBC methods .................................... 228
  - Unsupported methods ....................................... 230
  - Statement types ............................................. 236
  - Statements .................................................. 236
  - Prepared statements ........................................ 240
- Connection to IBM Cognos Virtual View Manager Server using ODBC ................................ 245
- ODBC client driver installation requirements .................. 245
Introduction

This document is intended for use with IBM® Cognos® Virtual View Manager.

This document assumes that you are familiar with the concepts underlying Virtual View Manager and the software's functionality as described in the IBM Cognos Virtual View Manager Getting Started Guide, a tutorial that illustrates a typical usage scenario.

Audience

This documentation is for information technology professionals who want to use Virtual View Manager to model data resources. Knowledge of relational data sources, hierarchical data sources, and data modeling will help you use this product.

Finding information

To find IBM Cognos product documentation on the web, including all translated documentation, access one of the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp). Release Notes are published directly to Information Centers, and include links to the latest technotes and APARs.

You can also read PDF versions of the product release notes and installation guides directly from IBM Cognos product disks.

Accessibility features

This product does not currently support accessibility features that help users who have a physical disability, such as restricted mobility or limited vision, to use this product. IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

The Great Outdoors Company, GO Sales, any variation of the Great Outdoors name, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder.
for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.
Chapter 1. IBM Cognos Virtual View Manager

This chapter describes the IBM Cognos Virtual View Manager user interface.

It assumes that you are familiar with the concepts described in the first chapter in the IBM Cognos Virtual View Manager Getting Started Guide.

These topics are covered:
- “IBM Cognos Virtual View Manager resource tree” on page 2
- “IBM Cognos Virtual View Manager menus” on page 7
- “Toolbar” on page 8
- “Editors and input dialogs” on page 9

IBM Cognos Virtual View Manager is the primary tool you use to access data sources and define, publish, and manage resources. Virtual View Manager lets you:
- Create, edit, and manage data sources, transformations, views, SQL scripts, parameterized queries, packaged queries, definition sets, triggers, Virtual View Manager databases, and Web services.
- Publish data sources, transformations, views, SQL scripts, parameterized queries, packaged queries, definition sets, triggers, Virtual View Manager databases, and Web services.
- Archive Virtual View Manager resources and deploy them back in a desired location with the export/import options.

Virtual View Manager serves as a data view design and development area. It has three basic functions: modeling, publishing, and resource management.

Modeling and publishing

These processes include several tasks such as data source introspection and metadata modeling, as well as creating Virtual View Manager databases and Web services.

To accomplish these goals Virtual View Manager provides three workspaces. The My Home and Shared workspaces are for modeling, and the Virtual View Manager Data Services workspace holds published resources.

For details on My Home, Shared, and Virtual View Manager Data Services, see “Desktop” on page 3.

Resource management

This process is multi-fold, and includes several tasks that you can perform on a resource, for example create, edit, save, move, copy, rename, export, import, delete, execute, and publish.

To manage resources, Virtual View Manager provides resource-right-click options, toolbar buttons, and menu bar options. Virtual View Manager also provides several editors with which you can define and refine the properties and parameters of a resource at anytime.
IBM Cognos Virtual View Manager resource tree

IBM Cognos Virtual View Manager displays the resources in a hierarchical structure, which is called the resource tree.

The resource tree is comprised of a root, container nodes (or, containers) and leaf nodes (or, leaves).

Resource tree hierarchy

A resource tree has four types of elements: container, leaf, parent, and child.

A container is a resource that contains other resources, which can be containers or leaves.

A leaf is a resource which does not contain any other resource.

A container that is not a leaf is a parent, and a resource that has a parent is a child. By this convention, only the root of the resource tree is not a child.

The notion of container, leaf, parent, or child is relevant to understanding the structure of the resource tree as well as the access control functionality, which is described in Chapter 9, “Security,” on page 153.

The term resource is generic and refers to any object that may be a container or a leaf. Therefore, this document uses the term resource when the notion of container/leaf/parent/child is not relevant.

Resource tree contents

The resource tree displayed in the left pane of Virtual View Manager is dynamically constructed. It displays only the resources the current user has permissions to view and may use.

When you launch Virtual View Manager for the first time after installation, the resource tree displays all the system-created containers. When you create a new resource to use in the system, you add that resource to one of these system-created containers or to another container that you have added to a system-created container. None of the top-level system-created containers can be edited or deleted.

This document uses the term resources to collectively refer to the resources that are used for data modeling and building business solutions using Virtual View Manager. These resources are data sources, views, parameterized queries, SQL scripts, Java procedures, packaged queries, transformations, and Virtual View Manager data services (which are available as Virtual View Manager databases and Web services). Data stored in these resources are available in tabular or hierarchical format, and noted accordingly as either tabular data and hierarchical data.

Resource Names are the parent container path combined with the resource name is the unique identifier for the invocation and reference to any Virtual View Manager defined resource.

The system-created nodes in the resource tree are described next.
Desktop
The Desktop represents the current user's virtual work area; it is analogous to the desktop in a personal computer's user interface.

Convenience containers display on the Desktop that refer to containers that are actually deeper in the server's resource hierarchy (Virtual View Manager Data Services, My Home, Shared), and they are on the Desktop to provide easier access to these other containers. The fourth container on the desktop, <Host Name> (localhost, if you are connected to Virtual View Manager on your local computer), actually represents the complete resource hierarchy of the server, and the contents of the other three Desktop containers can be alternatively found somewhere within the <Host Name> container (see “Relationship between resources in the resource tree” on page 6).

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

Virtual View Manager Data Services
Virtual View Manager Data Services is a system-created container that cannot be edited or deleted.

It has two system-created containers, Databases and Web Services, into which you publish the resources that you want to make available to client applications that use JDBC/ODBC/SOAP to connect to the Virtual View Manager Server.

Specifically, you publish the resources you want to make available to JDBC/ODBC clients into Databases, and the resources meant for SOAP clients into Web Services.

For details on publishing, see Chapter 8, “Resource publishing,” on page 147.

The published resources are the ones that the server exposes to JDBC/ODBC/SOAP clients.

For details on how JDBC/ODBC clients can access the published resources, see Chapter 12, “Client interfaces,” on page 223.

Databases and Web Services within Virtual View Manager Data Services display all the resources published by any user in the system. Each published resource in this area refers to an unpublished resource residing elsewhere in the system.

Only Databases and Web Services can exist directly within Virtual View Manager Data Services, so you cannot add any resource directly to Virtual View Manager Data Services. However, you can add new containers to hold published databases and web services to the original Databases and Web Services, which are described next.

Databases:

Databases is a system-created container that cannot be edited or deleted. This is the container to which you add your Virtual View Manager databases that are visible to JDBC/ODBC clients. The same set of Virtual View Manager databases residing here is visible to every client regardless of login status. That is, all users in the system share the same set of Virtual View Manager databases. However, users must have the right set of access privileges to use these databases.
To organize your Virtual View Manager databases, you can add resources such as catalogs to a database and a schema to a catalog.

For details on how to add Virtual View Manager databases, catalogs, and schemata, see Chapter 2, “Resource management basics,” on page 11.

**Databases > system:**

The Databases > system node contains system tables that are used by Virtual View Manager. The system tables are subject to change with new releases of the system.

For details on system tables, see the Virtual View Manager System Tables in the IBM Cognos Virtual View Manager Reference Guide.

**Databases > examples:**

The Databases > examples folder contains a sample view that has been published for clients who access the Virtual View Manager server via JDBC/ODBC.

**Web Services:**

Web Services is a system-created container that cannot be edited or deleted. This is the container to which you add your Web services that are visible to a SOAP client. The set of Web services available here is shared by all users that have the right set of access privileges to use these Web services. Any number of Virtual View Manager data services can be created in this container.

**Web Services > system:**

The system node within Web Services contains built-in Web services such as admin and util. Each of these Web services contains one or more services, ports, and operations.

Expand the nodes to view the available services, ports, and operations. The operations are the system Web service operations.

To obtain the details on any operation in these services, double-click the operation name, and click the Info tab in the lower section of the editor that opens on the right.

**My Home**

My Home is a semi-private work area (semi-private because everything is visible to an Administrator with a full set of rights) where sources may be designed, developed, and tested prior to exposure as either shared resource definitions or published as externally available resources.

The My Home folder is the user home folder or home area and as a system-created container it cannot be edited or deleted. The My Home folder is a shortcut provided for convenience of the user. The My Home folder and the corresponding [localhost]/users/Domain/CurrentUser folder are identical in every respect.

Add new instances of any of the following resource-types directly to My Home:

- Data Source
- Definition Set
- Folder
These resources may make reference to other shared or published resources. For details on adding and publishing resources, see Chapter 2, “Resource management basics,” on page 11 and Chapter 8, “Resource publishing,” on page 147.

**Shared**

Shared is a system-created container that cannot be edited or deleted. It is shared by all users in the system and is also intended as a location to store projects that are worked on by teams of people.

All the resources contained in the Shared area are accessible to users with appropriate access privileges. For details on access privileges, see “Privileges” on page 159.

**Shared > examples:**

The examples folder within Shared is a sample folder with a few sample resources to help you have a quick start at using Virtual View Manager.

**Host Name**

Host Name is the name of the Virtual View Manager server to which you are connected. If you are connected to server on the local computer, this name is localhost by default.

Host Name is a system-created container that cannot be edited or deleted. It represents the entire Virtual View Manager server on your machine. Host Name has four system-created containers—lib, services, shared, and users, which are described next.

**lib:**

This container is system-created and cannot be edited or deleted. It represents the Virtual View Manager library and contains all the built-in procedures in the system. These procedures can be published as Virtual View Manager data services and accessed from client programs.

For details on these procedures, see the “Built-In Procedures” in the *IBM Cognos Virtual View Manager Reference Guide*.

To view the information in the *Info* panel for each built-in procedure, double-click the desired procedure name, and click the *Info* tab in the lower section of the editor that opens on the right.

**services:**

This container is system-created and cannot be edited or deleted. It is the same folder as the desktop folder named Virtual View Manager Data Services.

For more information, see “Virtual View Manager Data Services” on page 3.
shared:

This container is system-created and cannot be edited or deleted. It is the same folder as the desktop folder named Shared.

For more information, see “Shared” on page 5.

You can add any resource to Host Name/shared, and the change is reflected in the structure of Desktop/Shared.

For further details on the correspondences between the resources in the resource tree, see “Relationship between resources in the resource tree.”

users:

This container is system-created and cannot be edited or deleted. It has one container for each security domain in the server. The default domain cognos is system-created, and cannot be edited or deleted. The system-created user is admin belonging to the domain cognos.

Each domain in users is represented by a folder-container, and each domain has a container per user in that domain, and this container is referred to as the user’s home folder.

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 username and resources belonging to the new user are displayed here.

For details on adding users to the system, refer to the IBM Cognos Virtual View Manager Administration Guide.

From users, you can view other users in the system and use their resources provided you have the right set of access privileges.

If you view your home folder from users, you will see your Desktop> My Home, because My Home represents your home folder.

Relationship between resources in the resource tree

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

<table>
<thead>
<tr>
<th>Resource in the workspace and published areas</th>
<th>Resource in &lt;Host Name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop &lt;current user&gt;</td>
<td>/users/&lt;domain&gt;/&lt;current user&gt;</td>
</tr>
<tr>
<td>Desktop &lt;current user&gt;/Virtual View Manager Data Services</td>
<td>/services</td>
</tr>
<tr>
<td>Desktop &lt;current user&gt;/Virtual View Manager Data Services/Databases</td>
<td>/services/databases</td>
</tr>
<tr>
<td>Desktop &lt;current user&gt;/Virtual View Manager Data Services/Web Services</td>
<td>/services/webservices</td>
</tr>
<tr>
<td>Desktop &lt;current user&gt;/My Home</td>
<td>/users/&lt;domain&gt;/&lt;current user&gt;</td>
</tr>
<tr>
<td>Desktop &lt;current user&gt;/Shared</td>
<td>/shared</td>
</tr>
</tbody>
</table>
IBM Cognos Virtual View Manager menus

The menu bar in IBM Cognos Virtual View Manager contains several menu options that are described in this section.

File menu

Offers the following menu options for data manipulation activities, change password, and switch user.

- New — creates a new resource.
- Open — opens the editor/information tab for the selected resource.
- Reopen — lists the most recently opened resources, which you can select to re-open.
- Export — archives the selected resource(s) into an archive file (*.car). This option is functionally similar to the pkg_export command-line program.
- Import into — uploads an archived file of saved resources. This option is functionally similar to the pkg_import command-line program.
- Close Active Editor — closes the editor for the resource that is currently being used/edited.
- Close All — closes all the open editors, active or not.
- Save — saves the current state of the selected resource.
- Save As — saves the current state of the current resource under a specified name.
- Save All — saves the current state of all the resources that are open.
- Refresh <resource> — synchronizes the state of the resource in the tree with its status in the server; displays incremental changes to the resource tree.
- Refresh All — synchronizes the state of all the resources in the tree with their status in the server; displays incremental changes to the resource tree.
- Change Password — opens the editor for changing a password.
- Switch User — logs out the current user and opens the log-in window to log in as a different user.

Edit menu

Offers menu options to edit resources: Undo, Redo, Cut, Copy, Paste, Delete, Rename, Find/Replace, and Go To Line.

For more information, see "Toolbar" on page 8.

Resource menu

Offers menu options for resource management tasks.

- Find Resource — Search for resources defined in the resource tree with a word fragment. Type a few letters and then wait a moment for a search response.
- Add/Remove Resources — lets you explicitly remove already introspected resources of a data source and/or introspect new resources from a data source.
- Enable Data Source — makes a data source available for use.
- Enable Trigger — makes a trigger available for use.
- Publish — opens the window for publishing a resource.
- Show Contents — displays the contents of a relational table.
- Show Dependencies — opens the Dependency panel which displays current view’s dependency.
• Test Service — opens your browser to test the connection to a Web service.
• View WSDL — opens your browser to view the WSDL of a Web service.
• Privileges — opens the window for granting access privileges to users and groups.

**Administration menu**

This menu is available only to users with administrative privileges. It provides menu options to add users, groups, and domains, configure system settings, and add custom hooks and custom functions.

• **Launch Administrator (Web)** — opens a Web browser to allow you to perform administrative tasks. Virtual View Manager Administrator allows you to monitor and manage licenses, cached resources, data source status, request status, and the status of sessions, transactions, and triggers. You can also manage domains, groups, and user membership and rights, and the view the event log. Some displays are granted to users with the right to View All Status though modification of settings is restricted with only that right. See the *IBM Cognos Virtual View Manager Administration Guide* for more information.

• **Change Owner of <resource>** — lets you change the owner of the selected resource so that a different owner may grant and manage privileges for the resource. Updates the privileges of the child and subordinate resources with the same owner change.

• **Full Server Backup** — opens a window for performing full back-up of the server. This option is functionally similar to the backup_export command-line program.

• **Configuration** — opens a window for setting system-level configurations for resources.

• **Custom Functions** — Opens a Custom Functions window to enable an administrator to promote or demote existing procedures that have a single scalar output and zero to many inputs. Custom functions are made available to all Virtual View Manager developers for easy inclusion in a column or as a criteria for inclusion from the grid view of any SQL based resource.

• **Server Logs** — creates a zip archive of all the logs. The logs are saved to a single file in the location you specify.

**Toolbar**

IBM Cognos Virtual View Manager contains buttons for tasks that all users can perform.

<table>
<thead>
<tr>
<th>Button</th>
<th>Menu option</th>
<th>Use to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Open icon]</td>
<td>Open</td>
<td>Open the editor/information tab for the selected resource</td>
</tr>
<tr>
<td>![Save icon]</td>
<td>Save</td>
<td>Save the current state of the selected resource.</td>
</tr>
<tr>
<td>![Refresh All icon]</td>
<td>Refresh All</td>
<td>Synchronize the state of all the resource in the tree with their status in the server; displays incremental changes to the resource tree.</td>
</tr>
</tbody>
</table>
Editors and input dialogs

IBM Cognos Virtual View Manager provides editors and input dialogs for creating/defining, annotating, viewing, or editing resources.

These resources are:
- folder, catalog, schema, Web service, port
- data source
- definition set
- procedure (SQL scripts, Java procedures, transformations, packaged queries, parameterized queries)
- trigger
- view
- Virtual View Manager service (Virtual View Manager database, Webservice). In addition to the editors that you can use to change the definition or content of a resource, there are property editors that you can use to perform the following tasks:
  - grant/revoke access privileges
  - add/remove resources from a data source
  - re-introspect a data source
  - gather statistics on a data source
  - set cardinality on tables and procedures in a datasource
- schedule the execution of a procedure or a view's SQL
- cache a view
- rebind a view or procedure
- publish a resource

The editors and dialogs are described in the sections in this manual where specific tasks are described.
Chapter 2. Resource management basics

This chapter assumes that you are familiar with the concepts described in the preceding chapter.

About resource management

Resource management involves creating, saving, modifying, moving, copying, and viewing user-created resources.

For advanced-level tasks of managing the contents of a resource—such as those included in the right-click menu options for a resource—see the chapter that discusses the resource.

Keep these things in mind:

- If you want to make a resource available to client programs through a JDBC/ODBC/SOAP connection, you must publish that resource. For details on publishing resources, see Chapter 8, “Resource publishing,” on page 147 in this guide.
- APIs are available for resource management. For a list of APIs for managing the resources, see /services/webservices/system/admin/resource in IBM Cognos Virtual View Manager. See also “Web Services > system” on page 4.
- Built-in procedures are available for managing resources in the system library in /lib/resource. For details on built-in procedures, see lib in this guide and the "Built-In Procedures" chapter of the IBM Cognos Virtual View Manager Reference Guide.

Resource creation

Creating a resource means that you are adding a new instance of the resource to the metadata repository.

The process you follow and the resource access rights depend on the type of resource:

- Creating a folder, catalog, schema, web service, or a port is a one-step process where you supply a name for the resource in an input dialog. For details on creating a folder, catalog, schema, web service, or port, see "Creating a folder, catalog, schema, web service, or port" on page 12.
- Creating a data source, definition set, procedure, trigger, or a view is a multi-step process. It involves using a specific editor and supplying the properties of the resource to be created. For details on creating a resource via an editor in IBM Cognos Virtual View Manager, see chapters 3-7.
- By default, only you and an administrator can use the resource you create. To share a resource with others, you need to grant access privileges on that resource. For details on access privileges, see "Privileges" on page 159.

Where can you create a resource?

You can create a resource in any suitable container on which you have the WRITE privilege.
You can create a resource in your home folder (My Home), the Shared node, or in someone else's home folder provided you have the privilege to write into the suitable container.

To find a suitable container for your resource, right-click the desired container and look for the New resource menu option; or, select the desired container and look for the File > New > resource_type menu option. If the option is listed and enabled, you can create your resource in that container.

Creating a folder, catalog, schema, web service, or port
Follow the steps below to create a resource.

Procedure
1. Right-click the container where you want to add the resource, and select New resource_type. Alternatively, select the container and then File > New > resource_type.
   The Input window displays.
2. Type a name for the resource in the input field, and click OK.

Creating a Virtual View Manager database or web service
Follow the steps below to create a Virtual View Manager database or web service.

You create Virtual View Manager databases within Virtual View Manager Data Services > Databases, and web services within Virtual View Manager Data Services > Web Services.

Procedure
1. Right-click Databases or Web Services, as needed, and select New Data Service.
2. Type a name for the data service (a database or web service) in the Data Service Name field, and click OK.
   In the resource tree, a newly created web service is placed within Virtual View Manager Data Services > Web Services whereas a newly created database is placed within Virtual View Manager Data Services > Databases.
   If you created a web service, you must also create a folder inside it, because you cannot publish directly to a web service. For information about creating a folder, see “Creating a folder, catalog, schema, web service, or port”.

How to open resources

You can open the editor for any existing resource in the resource tree. How you do this depends on the type of resource.

To open most any resource (except folders or data sources)
Right-click and select the menu option to Open the resource, or simply double-click the resource name in the resource tree.

To open a folder or data source
Right-click the folder or data source and select Open to open the resource editor.
Resource locks and editing resources

When resource locking is enabled resources must be locked so that edits may be made.

Resource locking prevents modifications made by another developer inadvertently overwriting changes on top of a resource that is currently being edited. Resource locking is not enabled by default, so edits may be made to any unlocked resource for which the user has the requisite privileges.

In either case, a resource may be opened, locked, and modified starting with a right-click on the resource name in the IBM Cognos Virtual View Manager resource tree.

When a resource is locked, other developers working on the same Virtual View Manager server instance will be able to see that a lock has been placed on the resource or resource tree node. Other users attempting to acquire a resource lock on a resource or tree that is already locked by someone else will be denied.

To break a resource lock held by some other developer, see "Access to locked resources" on page 14

New resources in locked containers

Only the user who locked a resource may modify and save changes on that resource. Only the lock owner may create or destroy resources within a locked container.

All resources within a locked container receive an implicit lock from the parent container that belongs to the original lock owner.

About locked resources

Locked resources may be quickly identified in the Virtual View Manager resource tree by the small gray lock next to the regular icon. All Virtual View Manager instances connected to the same server will see the same set of locked icons for those resources for which they have Read privilege access.

In the graphic below, the locks are highlighted in yellow. Information about resources that are locked may be obtained by simply hovering the cursor over the resource name.

Figure 1. Example of a locked resource

For more detailed information open the resource and select the Info tab. The Info tab shows whether or not the resource is locked and other lock related information:

- **Lock owner** — user who set the lock
• **Lock owner domain** — domain of the user who set the lock
• **Lock creation time** — when the lock was set by the user
• **Lock type** — **Explicit** (user specifically set a lock on the selected resource) or
  **Implicit** (lock is imposed by a lock on a parent container)

**Locking change history**

The server events log captures a change history for resources that are locked and unlocked.

The file named cs_server_events.log contains events that capture locking and unlocking events with the user name and domain of the designer who locked or unlocked the resource, the timestamp, the data resource, and any descriptive annotation the designer entered at the time of unlocking.

Change events may also be monitored with a third party SNMP utility. SNMP IDs 20805 (csResourceLock) and 20806 (csResourceUnlock) will reveal locked resources and changed resources with change descriptions entered by the designer when the top most resource node was unlocked.

Refer to the “Server Event Attributes” topic in the “System Monitoring” chapter of the *IBM Cognos Virtual View Manager Administration Guide* for more information on SNMP monitoring.

**Access to locked resources**

Other users are prevented from making significant changes (saved changes) to locked resources until the resource is either unlocked by the lock owner or unlocked by an administrative user with the Unlock Resources right.

Resource locking is not meant to be a security feature for use in permanently securing resources. An administrator may better protect defined resources by managing rights and privileges granted to groups and users for use of a set of published resources.

Administrators with the Modify All Users right and resource owners may revoke the Write privilege (modify) for specific resources to prevent modification of those resource definitions.

**How to unlock resources**

Locked resources may be unlocked by the lock owner, or they may be unlocked by a user/administrator with the Unlock Resources right. Resources should be saved prior to unlocking, especially when locking is enabled.

A right click on any locked resource to display the contextual menu option to Unlock Resource. The Unlock Resource dialog box enables unlocking of groups of locked resources. All resources that may be unlocked by the current user are shown. It is most advisable to unlock an entire tree of resources instead of individual resource to free all contained objects simultaneously. If the user has the Unlock Resources right then all other locked objects for which the user has Read privilege will be shown and selected for unlock as may be required.
How to edit resources

When resource locking is enabled, any user who wants to change an existing resource is required to acquire a lock on either that resource or on the parent container prior to modifying and saving any of those resources.

If a lock is already set on a container or a resource, other users attempting to acquire a resource lock will be denied.

Different types of resources may be modified in different ways. Editing a resource means performing one or more of the following tasks, depending upon the resource:

- **Annotate any type of resource (except the Desktop node) with a note.** See "How to annotate a resource."
- **Modify the definition, including connection credentials in the case of a data source.** See Chapter 3, “Data sources,” on page 27
- **Modify the selection of tables, column projections, or SQL in the case of a view.** See Chapter 4, “Views,” on page 75
- **Modify the SQL and parameters in the case of a transformation, packaged query, or SQL Script.** See Chapter 5, “Procedures,” on page 91
- **Make changes to definitions.** See Chapter 6, “Definition sets,” on page 127

How to annotate a resource

Notes (annotations) may be added to any resource if you have read access and Write privilege on the specific resource.

Annotations may be seen and modified from the Info tab of any opened resource.

How to save resources

The same resource locking protections that protect against simultaneous modification of locked resources constrain the saving of resources. If a resource is locked, only the lock owner may save changes to that resource.

A locked resource may be opened and saved with a different name in a different folder so that parallel development may be progress on similar objects.

Saving a resource means saving the changes made to the definition, parameters, annotation, or other properties of the resource in the metadata repository.

**Saving a resource under the same name**

Follow these steps to save a resource under the same name.

**Procedure**

1. Edit the desired resource, if necessary. See “Moving a resource” on page 16
2. Do one of the following:
   - Click the Save button on the toolbar.
   - Select the File > Save resource option.

**Saving a resource under a different name**

Follow these steps to save a resource under a different name.
Procedure
1. Edit the resource if necessary. See “Moving a resource.”
2. Use the File > Save As option.

Saving multiple resources
Follow these steps to save multiple resources.

Procedure
1. Edit the desired resources, if necessary. See “Moving a resource.”
2. Use the File > Save All option.
   This function saves all resources that were modified on the desktop.

Moving a resource
You can move a resource by cutting a resource and pasting it into another location or into the same location from where you cut it.

The following rules exist for cutting and pasting resources:
• You cannot move an unpublished resource into Virtual View Manager Data Services, which only stores published resources, and vice-versa.
• You cannot move the children of a data source individually; you have to move the entire data source.
• You can move a container, and when you do all of its contents are moved.
• You can move a leaf, such as a view or a procedure.

The following things happen when you move a resource:
• When you cut a resource, the resource is put in the clipboard until a subsequent paste operation, but its reference still remains in its original location (from where you cut it).
• If you rename the reference immediately following a cut operation and then do the paste operation, the resource is pasted into the new location with its new name.
• If you cut a resource named A and then paste it, the resource is pasted with the name A in the specified location.
• If you cut a resource named A, then copy a resource named B, and then do a paste action, B is copied. The copy operation (of B) displaces the resource (A) placed in the clipboard by the cut operation.

Procedure
1. Right-click the name of the desired resource, and select Cut.
   Alternatively, you can first select the resource name, and then the Cut toolbar button.
   The cut resource is put on the clipboard.
2. Right-click the name of the desired container to place the cut resource, and select Paste into.
   Alternatively, you can first select the container name, and then the Paste into button on the toolbar.

Copying a resource
Copying a resource works in a way similar to moving a resource, except that you must first copy the resource and then paste it into the desired location.
**Procedure**

1. Right-click the name of the desired resource, and select **Copy**.
   Alternatively, you can first select the resource name, and then the **Copy** toolbar button.
   The resource to be copied is put on the clipboard.

2. Right-click the name of the desired container to place the resource in the clipboard, and select **Paste into**.
   Alternatively, you can first select the container name, and then select the **Paste into** button on the toolbar.

**Deleting a resource**

If you delete a container, all of its contents are deleted. When you delete a resource from the resource tree, the resource is removed from the metadata repository.

Accordingly, the deleted resource is removed also from other users' **Desktop** if they previously had access privileges to view and/or use that resource.

**Procedure**

1. Right-click the desired resource, and select **Delete**.
   Alternatively, you can select the desired resource and click the Delete button.
   To select multiple resources for deleting, press Ctrl and select the resources one by one, or press Shift and select a set of adjacent resources.

2. In the **Confirmation** window, click **Yes**.

**Renaming a resource**

Follow these steps to rename a resource.

**Procedure**

1. Right-click the desired resource, and select **Rename**.
   Alternatively, you can first select the desired resource and then the **Edit > Rename** option.

2. Place the cursor anywhere on the resource name, type a new name, and press Enter.

**Resource export and import**

All object resources in the IBM Cognos Virtual View Manager directory may be saved to an archive file to port those objects to another instance or to take a snapshot of a configuration for safeguarding.

Archiving a resource is called exporting and 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 may be redeployed by importing the CAR file. The CAR file is referred to as the export file or import file when backing up, porting, or restoring the server or saved Virtual View Manager objects.

The entire Virtual View Manager configuration (excepting local machine settings), a group of resources, or a single resource object can be exported. The user must have the appropriate access rights for the objects they are exporting. All objects can be exported by an administrator with all rights, or by a operations administrator with...
**Backup** (or **Backup and Restore**) rights who has **Access Tools**, **Read All Config**, **Read All Resources**, **Read All Status**, and **Read All Users** rights. Import requires the complementary **Modify** right of all those rights listed for export.

Full Server Backup may be invoked from Virtual View Manager’s **Administration** menu to create a single CAR file. The CAR file can be stored or used to restore the Virtual View Manager server to that same snapshot of the server. A CAR file may be used to port an entire instance or just selected objects to replicate or restore saved configurations to a Virtual View Manager server of the same release version.

You can export individual objects from the Virtual View Manager resource tree with a right-click on selected objects and using the **Export** option. The user performing the export must have **Read** privileges on the selected objects and other associated rights if all related object resources are to be exported too. The exported CAR file may include or exclude cache configuration, data source configuration, or Java Jar files that might be required. Export options also enable inclusion of privileges without any additional rights.

When the user performing the export has the **Read All Users** right, user profile definitions may be included (as in the pkg_export option -includeusers) in the export archive file. These items are disabled if they do not have this right.

Resources captured in a CAR file may be imported to a Virtual View Manager instance of the same release version if the user has **Write** privilege on the targeted import folder, or when importing to the folder specified in the CAR file.

Include privileges during import of a CAR file if the user performing the import has both the **Modify All Resources** and the **Modify All Users** rights and presuming the CAR file includes privilege information.

### Locked resource export and import

Different options are available to export and import resources.

#### Locked resource export (full server backup and backup_export)

All locked resource metadata is exported when using either the **Full Server Backup** feature available from the Studio Administration menu or the backup_export command line utility.

The CAR file produced by these export methods contains all pertinent lock metadata and depending on how the file is later imported those resources may be restored with the locks intact.

One may verify the presence of locking metadata in the archive (*.CAR) file exported as explicit locks. Locking metadata is present in the file called resourcelocks.xml within the zipped CAR file.

Export using either pkg_export or from the menu option available from the Virtual View Manager resource tree does not export lock metadata.

#### Locked resource import (backup_import)

Lock status and lock ownership are ported to the target Virtual View Manager server when a full server backup CAR file is restored using the backup_import command line utility.

By default the backup_import command line utility adds CAR file resources to existing resources in identical container paths. If both the resource paths and the
resource names are identical in the target server and in the CAR file, then the CAR file resource overwrites the target resource. Refer to the documentation on backup_import for more details on how resources are merged.

Specify the -overwrite option to clear existing resources from the target and to set the Virtual View Manager server to the resource definitions present in the full server backup CAR file.

**Locked resource import (pkg_import and Studio Explorer import)**
The pkg_import command-line utility imports the resources but not any lock metadata.

Likewise the **Studio > Import** option from the menu does not import lock metadata.

If the user performing the import has the **Unlock Resources** right then the import operation may opt to override locks applied to resources. When **Override Locks** is selected, the import will clear resource locks and overwrite resources with identical path names and identical resource names.

For more information on import and pkg_import options refer to the Utilities chapter in the *IBM Cognos Virtual View Manager Administration Guide*.

Virtual View Manager offers an editor through which you can export/import a resource if you have the Read/Write permission on that resource. This procedure has its limitations, which are described later in this section.

**Resource export**
Export from the Virtual View Manager resource tree creates a discrete CAR file with all the metadata of the selected object and any contained child objects present in the folder directory selected.

Selective object export works much like the command-line utility pkg_export default behavior with the -description option set. To learn more about the command line utility refer to the Utilities chapter in the *IBM Cognos Virtual View Manager Administration Guide*.

Resources selected to be exported are exported with all contained child objects.

**Exporting locked resources (pkg_export and Studio Explorer export)**
Export from the Virtual View Manager resource tree does not export locked metadata.

Export from the selected nodes and resources selected on the Virtual View Manager resource tree > Export option create a CAR file containing the selected resources that excludes the locked metadata.

The resource definitions selected for export are exported regardless of whether they are locked or not. The locked container and locked resource information is not present in exported CAR file.

**Procedure**
1. Right-click the desired resource in the resource tree, and select **Export**.
Alternatively, you can first select the resource and then File > Export <resource>.

To select multiple resources for exporting, press the Ctrl key and select the resources one by one, or press Shift and select a set of adjacent resources.

2. In the Export <resource> window, type a name for the CAR file or use the Browse button to locate a file to store the exported resource.

3. In the Include Resource Information section of the Export <resource> window, select and/or deselect export options as required.

Exported CAR files may include caching configurations as well as the cached data, data source connections, access privileges, custom jars, dependencies, cardinality statistics, or required user(s).

Some check boxes may not be available (grayed out) if the user profile being used does not have rights for the action provided by the option. For example the Privileges check box is enabled only for users with the Read All Users right.

Table 1. Export options and descriptions

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching</td>
<td>Includes resource caching configurations such as storage type and location, scheduling information and also materialized view data.</td>
</tr>
<tr>
<td>Data Source Connections</td>
<td>Includes resource specific connection profiles and configurations, including host, port, login, and password if provided. This option requires the Read All Resources right.</td>
</tr>
<tr>
<td>Privileges</td>
<td>Includes resource group and user privilege information. User privilege information for exported resources is imported into the target Virtual View Manager installation only if the exact same user name, group, and domains exist on the target and the user profile performing the import has either the Modify All Users right or the Modify All Resources right, else privilege information is not imported.</td>
</tr>
<tr>
<td>Custom Jar</td>
<td>Includes any custom jar created to support a new data source.</td>
</tr>
<tr>
<td>Include Dependency</td>
<td>Includes any dependency resources in the exported CAR file. View dependencies that will be exported by opening the resource and selecting the dependency panel.</td>
</tr>
<tr>
<td>Include Cardinality Statistics</td>
<td>Includes data source statistics when present, as well as configurations such as refresh mode, refresh time-frame, manual cardinality overwrites, and specific column setting when present.</td>
</tr>
<tr>
<td>Include Required Users</td>
<td>Includes the owner(s) of the resource(s) exported.</td>
</tr>
</tbody>
</table>

Choosing check box options in this section is equivalent to enabling options using the command line utility. Refer to the Utilities chapter in the IBM Cognos Virtual View Manager Administration Guide for more information.

4. Optional: Enter a description of the export file in the Description box which will be visible when the exported CAR file is readied for import.

5. Click OK.

**Marking of rebindables during export**

Many resources like views, procedures, and web services are dependent on one or more underlying sources. These resources are considered bound or dependent upon those sources as dependencies.
Imported resources often need to be re-bound to dependencies so that underlying sources properly feed the replicated resource in its new location. Binding a resource to its dependencies is often part of the import process. For example, a procedure P1 may 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 upon 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 instead of a QA/development data source.

For additional information, see “Rebinding a view” on page 89 and “Rebinding procedures” on page 119.

Exported resources are often bound to underlying sources, and the binding should be reestablished (or changed) during an import of the resources. The command-line export utility program (pkg_export) provides an option to mark the underlying source as rebindable using the command:

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

When a resource is marked as rebindable in the CAR file, Virtual View Manager import of this resource will display the path and description in a dialog box that enables specification of a new path for the dependency resource.

The command-line import utility program (pkg_import) provides another option for rebinding resources:

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

The following section describes the more simple mechanics of how to import a resource.

**Resource import**

You import CAR files into to replicate Virtual View Manager resources and resource configurations. CAR files are created by export of resources, and they usually contain many object resources or they may even contain a full server backup.

For details on creating a CAR file, see “Resource export” on page 19.

Resources imported from an archive get placed within the selected folder in the Virtual View Manager resource tree. Import performs a path relocation on all directory path roots in the archive to change resources from their old path to that of the currently selected folder on the target Virtual View Manager server.

All import rules apply (see “Rules for importing resources” on page 22).

The **Refresh** button works like -printinfo: the user gets to see information stored about the archive file.

The **Preview** button works like the -messagesonly option. The -messagesonly option displays the messages generated in a package import without actually performing the import. For more information on this option and others like it, see the documentation on pkg_export and pkg_import in the "Utilities" chapter in the *IBM Cognos Virtual View Manager Administration Guide*. 

Chapter 2. Resource management basics  21
In Virtual View Manager, you can first select a folder on which you have the Write permission, and then select the menu option File > Import into. Or, you can right-click at the folder and select Import. The dialog that comes up lets you choose the file and whether or not to include optional resource information.

**Rules for importing resources**
Importing follows some rules to resolve conflicts during import.

These rules are:

- If an imported resource does not exist, it is created. The person performing the import gets appropriate 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 -includeaccess, then the owner of the resource is set as it was before and any privileges in the import are also put in place.

- If a resource is imported to a non-existent folder, the folder (and any parent folders of that folder that don’t yet exist) is created and the user performing the import gets READ|WRITE privilege and ownership of these folders. Auto-creation of missing folders is not supported in the Virtual View Manager Data Services area.

- Import of a pre-existing resource results in overwrite of the old version. Overwrite is complete except for the following:
  - The owner is not changed. The original owner retains ownership.
  - User privileges are not overwritten unless explicitly changed by the import. For example, if abe has READ|WRITE and bob has READ|WRITE, and the import lists abe as READ but does not mention bob then abe’s privileges are updated but bob’s are left intact.
  - If the resource is a folder or data source, its children resources are not removed.

These restrictions apply to importing resources:

- The configuration settings (done through the Administration > Configuration menu option) are not carried over when you export/import a resource using.
- Resource creation in a folder requires WRITE privileges on the directory path.
- Resources may not be overwritten without WRITE privilege for that resource.
- You cannot export just part of a physical data source; you must export either all or nothing.
- Physical data sources must be imported as a whole, not partially. If you import, you must include the source definition itself.
- The Virtual View Manager Data Services area has strict structure rules that are enforced.
- You cannot import anything that was exported from the Virtual View Manager Data Services area to outside of that area.
- You cannot import anything that was exported from outside the Virtual View Manager Data Services area into that area.

**Importing a resource**
Follow these steps to import a resource.

**Procedure**
1. Right-click the desired container into which you want to import a resource, and select Import.
Alternatively, select the resource and then the option: **File > Import Into <resource>**.
The Import into <...> window opens.

2. Use the **Browse** button to locate and upload the desired CAR file.
   You can also type the full path to the import file in the File field.
3. Select or deselect import options in the **Include Resource Information** section as required.
   By default, some check boxes to import the details about a resource's caching, data source connections, privileges, and scheduling are selected. If the exported CAR file includes extraneous or undesired resource information those aspects may be omitted by deselecting the appropriate check box. If the export CAR file does not contain some resource information by virtue of exclusion on export, of course it is impossible to include that resource information by checking the respective box here.

   Selecting options in the **Include resource information** is similar to specifying the option `-includeaccess` in the pkg_import program.

   Some check boxes may not be available due to limits in user rights or because the selected CAR file does not have those objects or options available for import. For example the **Privileges** check box is enabled only for users with the **Read All Users** right.

   **Table 2. Import options and descriptions**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caching</td>
<td>Includes/excludes resource caching configurations such as storage type and location, scheduling information and also materialized view data.</td>
</tr>
<tr>
<td>Privileges</td>
<td>Includes/excludes resource group and user privilege information. User privilege information is imported into the target Virtual View Manager installation only if the exact same user name, group, and domains exist on the target and the user profile performing the import has either the Modify All Users right of the Modify All Resources right, else privilege information is not imported.</td>
</tr>
<tr>
<td>Overwrite</td>
<td>Overwrites/or aborts import of resources that have both identical names and identical path directories. Owner and user privileges are not overwritten unless they are explicitly changed by any new privilege setting.</td>
</tr>
<tr>
<td>Custom Jar</td>
<td>Includes/excludes any custom jar created to support a new data source.</td>
</tr>
<tr>
<td>Data Source Connections</td>
<td>Includes/excludes resource specific connection profiles and configurations, including host, port, login, and password if provided. This option requires the Read All Resources right.</td>
</tr>
<tr>
<td>Include Users</td>
<td>Imports/excludes the owner(s) of the resource(s) when present in the CAR file. When the Users are omitted, the user performing the import becomes the owner of the resource.</td>
</tr>
</tbody>
</table>

   Choosing check box options in this section is equivalent to enabling options via command line in the pkg_import utility. Refer to the Utilities chapter in the *IBM Cognos Virtual View Manager Administration Guide* for more information.

4. 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 a part of the server or the entire server.
   - **Date**—Day, date, and time when the resource was exported.
• **User**—user that exported the resource.
• **Server**—host machine and port from where the resource was exported

5. Select the **Show Rebinding Options** box for binding a resource to an underlying source for the first time.

6. To view a list of the contents of the import file without actually performing any imports, click the **Preview** button.

7. Click **Import** to import the resources.

8. Click **Done** to complete the import process.

---

**Using IBM Cognos Virtual View Manager for a full server backup**

IBM Cognos Virtual View Manager offers the capability to perform a full server backup.

This capability is available only for those with these administrative rights: **Access Tools**, **Read All Resources**, **Read All Users**, and **Read All Config**.

This option is functionally similar to the backup_export command-line program.

**Procedure**

1. Select the **Administration** > **Full Server Backup** menu option.
   
   The **Full Server Backup** window opens.

2. Click the **Browse** button to specify a file to store the exported resource.
   
   The **Save** window displays for specifying/locating an export file. You can either:
   
   • Create a new export file by typing a name for it in the **File name** field.
     
     You don’t need to specify the filename extension (CAR) since it is automatically added.

   • Use an existing CAR file; locate it through **Browse** and click **Save**.
     
     Note that if you use an existing CAR file, its contents will be overwritten when you complete the export process.

3. Type a description for the export file in the **Description** box, if required.

4. Click **OK**.

---

**Searching for resources in the server**

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

**Procedure**

1. Select the **Resource** > **Find Resource** menu option.
   
   Alternatively, you can press the keyboard combination Ctrl + N.

   The **Find Resource name** window opens.

   If you want to close the search window, click anywhere outside the window, or press the escape key (Esc). Clicking the search window’s title bar will not close the window.

2. In the text field, type the name of the resource you want to find.
   
   You can use the following wild card characters:
   
   • a period (.), to match a single character
   • an asterisk (*) or a percentage symbol (%) to match any number of characters
A timer starts and waits for you to stop typing. This is determined using a short timeout of around 300 milliseconds. Each time you type a character, the timer is restarted.

When you stop typing, the server is queried for any resources whose name begins with the text you typed, and resource references containing the resource name, path, and type are returned.

The resource references are displayed in a drop down window below the text field where you entered the search text. The type of the resource can be identified by the icon preceding its name.

3. Use the up/down arrows to navigate through the list of resources.

4. If you do not see the desired choice, restart typing, including the use of backspace and the left and right arrow keys to change the search pattern.

When you pause, the search commences and results are displayed like before.

5. Repeat Step 3, and select the desired resource either by clicking on it or by pressing the Enter key.

The resource opens on the right. Simultaneously the resource tree (on the left) expands to display the location of the resource in the tree.

### Viewing table schema and contents

You can view a table's schema and contents even before including it in a view.

Viewing the contents of a table would be affected if the table has column-based security (see "Privilege enforcement on sources underlying a resource" on page 164 for more details).

**Procedure**

In the resource tree, right-click the desired table and select **Show Contents**.

The table view panel opens on the right, displaying the table columns and their data types. The **Result** panel displays the data contained in the table.

### Resource publishing

If you want to make your tabular data and procedures available to JDBC, ODBC, ADO.NET, and SOAP clients, you should publish them as IBM Cognos Virtual View Manager data services.

For details on publishing a resource, see Chapter 8, “Resource publishing,” on page 147.
Chapter 3. Data sources

This chapter describes how to discover the data sources existing on the network, add new data sources and data source drivers, and re-introspect data sources.

This chapter assumes that you are familiar with the basic concepts of IBM Cognos Virtual View Manager and the features discussed in the preceding chapters.

Supported data source types

IBM Cognos Virtual View Manager supports different data sources.

The supported data source types are:
- Relational, see “Relational data sources” on page 29
- LDAP, see “LDAP data sources” on page 45
- WSDL, see “WSDL data sources” on page 45
- XML/HTTP, see “Adding XML/HTTP data sources” on page 48
- File Type: cache, delimited, or XML, see “File data sources” on page 52
- Custom Java procedure, see “Custom Java procedures” on page 55
- ERP data sources, see “ERP data sources” on page 61

Note: If you want to make a data source available to client programs through a JDBC/ODBC/SOAP connection, you must publish that data source.

For details on publishing resources, see the chapter Chapter 8, “Resource publishing,” on page 147.

Auto-discovering data sources

You can automatically discover existing relational data sources at particular ports on your network, and add the desired data source to the server.

Procedure
1. Right-click at an appropriate location in the resource tree, and select New Data Source.
2. In the Auto Discovery section of the Add Physical Data Source window, click Auto Discovery.
   The window 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 machine from which to scan the network, and click Scan Network.
   This command provides an entry point to the network, and the server discovers a set of host machines where specific data sources exist.
4. Select the desired machine name (or IP address), and click Next.
   For further details, refer to “Relational data sources” on page 29.

What it means to add data sources

Adding a data source means creating a representation of the underlying data source in the IBM Cognos Virtual View Manager metadata repository for querying.
You can add an entire data source or be selective about the data source resources — such as catalogs, schemas, tables, stored procedures — you want to add. The data source metadata together with the details for connecting it to the server formulate the Virtual View Manager representation of the data source. Provided you have the right set of access privileges, you can add a data source anywhere in a user’s home folder (Desktop/My Home) or in the shared area (Desktop/Shared).

**Data source wizard**

Virtual View Manager has a wizard for adding data sources to the server.

You can launch this wizard by one of the following methods:

- Right-click at an appropriate location in the resource tree, and select **New Data Source**.
- From an appropriate location in the resource tree, choose **File > New > Data Source**.

When you follow one of the preceding steps, the *Add Physical Data Source* window opens. This window lists the drivers available for connecting your data source and the Virtual View Manager server.

**Introspection and re-introspection**

The task of adding a data source to the metadata repository is called *introspection*. *Re-introspection* is the process of looking for differences between an introspected data source and the underlying data source.

**Introspecting data sources**

Generally, the steps you go through to introspect a data source are similar for all data sources. While there are specific differences for each data source type as described in the rest of this chapter, you introspect data sources using this process.

**Procedure**

1. Specify the driver to connect to the desired underlying data source.
   The server will note your choice of the driver.
2. Provide relevant information about the underlying data source so that the Virtual View Manager server can connect to it.
   The server validates the connection details, and gives an error message if the validation fails.
3. Select from the underlying data source the resources you need for your data modeling.
   The server displays the names of all the resources in the underlying data source for you to select.
4. Direct the server to add the data source.
   The server fetches the resources you selected in the previous step and adds them to the metadata repository.

**Re-introspection process**

Large data sources are likely to contain voluminous data distributed in numerous tables, some of which may not be relevant to your modeling needs. Instead of forcing you to upload the entire data source into its metadata repository, Virtual View Manager lets you select (during introspection) only those resources that you need for your modeling purpose.
If there is a change in the underlying data source later on, you can use the Virtual View Manager server to detect such changes.

Re-introspection is the process of looking for differences between an introspected data source and the underlying data source.

For further details, see “Data source re-introspection” on page 57.

**Relational data sources**

For the relational data sources listed below, the connection properties and the procedure for adding an IBM DB2® data source is essentially the same for all relational data sources.

First refer to the section on “Adding an IBM DB2 data source” on page 30 for more information.

Supported relational data sources are as follows:

- IBM Cognos Virtual View Manager Data Services
- Custom Java Procedures
- IBM DB2 Type 2 and Type 4
- IBM DB2 z/OS® Type 4
- IBM Informix®
- Microsoft SQL Server
- MySQL
- Netezza®
- Oracle w/ Thin or OCI Driver
- Sybase
- Teradatas

Specific connection details, parameters and settings are slightly different for each data source, so consult the data source specific section that describes how to add the following types of relational data sources to the Virtual View Manager server:

- “Process to add an IBM Cognos Virtual View Manager Data Source” on page 35
- “Custom Java procedures” on page 55
- “Adding DataDirect Mainframe data sources” on page 35
- “Adding an IBM DB2 data source” on page 30
- “IBM Informix data sources” on page 38
- “Microsoft Access data sources” on page 44
- “Microsoft Excel data sources” on page 45
- “Microsoft SQL Server data sources” on page 39
- “MySQL data sources” on page 40
- “Netezza data sources” on page 40
- “Oracle data sources” on page 40
- “Sybase data sources” on page 42
- “About adding Teradata data sources” on page 43

The configuration dialogs for adding a Microsoft Access or a Microsoft Excel data source are also similar to one another. Please refer to the respective section for configuring those data sources for use with Virtual View Manager.
The following sections describe how to configure the server to connect with these data sources.

**Procedures for adding relational data sources**

The procedure for adding an IBM DB2 data source to the metadata repository is described in the next section. Because the process of adding other relational data sources is similar except for the connection details, the specific differences for each other relational data source are described in the following sections.

**Adding an IBM DB2 data source**

You can add IBM DB2 (Type 2 and Type 4) data sources, and mainframe IBM DB2 z/OS (Type 4) data sources.

You can manually install the JDBC driver that will connect your IBM DB2 data source to IBM Cognos Virtual View Manager Server.

For details on obtaining the JDBC drivers for IBM DB2 and installing them, refer to the "Installing Special JDBC Drivers" topic in the "Post-Installation Tasks" chapter of the *IBM Cognos Virtual View Manager Administration Guide*.

**Procedure**

1. Right-click at an appropriate location in the resource tree, and select New Data Source.
2. In the Add Physical Data Source window, select a driver, and click Next.
3. In the Add Physical Data Source window for provide connection information and other details, click the Basic tab if necessary, and then supply the relevant information in the following fields:
   - **Name** — User-supplied name to represent the underlying data source. When the process of adding this data source is complete, the user-supplied name will be displayed in the resource tree.
   - **Host** — Name of the machine hosting the Virtual View Manager server or the host machine's IP address.
   - **Port** — Port number for the data source to connect with the host.
     For IBM DB2 v8 Universal Database, the default port number is 50000.
     For IBM DB2 z/OS8 (Type 4) data source, the port number is 446.
   - **Database Name** — Name of the underlying data source. For IBM DB2 Type 2, this is database alias name.
     The Virtual View Manager server uses this name to find and connect to the underlying data source.
   - **Login and Password** — Valid user name and password that are required to access the underlying data source.
   - **Save Password** — By default the login and password are saved because it is very desirable to have a reusable system connection pool with the data source. The data source connection pool may only be used by properly authorized groups and individuals explicitly given privileges to select or execute queries and procedures on that source.
     When the password is saved (default setting), the resource owner, explicitly authorized groups and users, or administrators may do the following without manual re-submission of the password:
     - Introspect the current data source
     - Re-introspect the data source
- Add/Remove data source resources
- Perform query/update/insert operations on a table in the data source
- Invoke a stored procedure
- Refresh a cached view based on the data source resources
- Use the query optimizer for statistics gathering

To disable saving of the password, enable **Pass-through Login**.

Virtual View Manager does not support multiple (pass-through) passwords for a single user. Currently each user session may utilize only one user login and password to create new connections with data sources enabled with pass-through login. If pass-through login is enabled on multiple data sources, then connection authorization must be based on a single user login and password pair for each user or connection to the data source will fail for federated queries.

- **Pass-through Login** — Disabled by default to allow automated provisioning of a connection pool. Open connection threads may be used by any properly authorized users after the validation query verifies connection status. Normally the Virtual View Manager Server maintains a minimum number of connection threads because the performance cost of establishing new connections is significant. Pass-through Login setting dictates availability of the Save Password check box.

When **Pass-through Login** is enabled, a new connection to the data source must be negotiated prior to the first request to select or execute a query or procedure in a given user session. Subsequent requests or executions sent to the same data source by that same user reuse the existing connection as a restricted basis for connection pooling. Since each pooled connection is established with a specific username and password, that connection can only be re-used by that specific user.

When a new user attempts to connect to a data source configured to use pass-through login a new connection is created if the maximum number of pooled connections has not been reached, or a stale connection will be dropped to create a new connection for that user.

Virtual View Manager Server supports pass-through login using a single user name and password for each user session. Pass-through login may be enabled for more than one data source if the same set of user names and passwords may be used to create data source connections.

- When the pass-through login is enabled (**pass-through mode**) and the password is saved, Studio mediated introspection may be performed without re-submission of the password.
- In pass-through mode, when the password is not saved:
  - Query, Update, Insert, and Delete operations require that the end-user re-supply the password for the current session.
  - Re-introspection, Add, and Remove data source resources requires use of the same password that was used when the data source was originally introspected.
  - Scheduled re-introspection can **not** be performed.
  - When a cache table is stored on a pass-through enabled data source scheduled cache refresh can only occur if the resource owner/creator credentials used for accessing both Virtual View Manager and the target data source using pass-through are the same. If this is not the case the cache refresh will fail.
  - Statistics gathering, using the query optimizer can not be performed.
If a view uses a data source that was originally added to Virtual View Manager server using the pass-through mode without saving the password, row-based security may affect the cache refresh functionality. For example, suppose that a cached view, named CachedCommonView, uses the SQL statement SELECT * FROM db2.T1; and the user John with the password john1 is allowed only to view 10 rows, whereas the user Jane with the password jane1 is allowed to scan 20 rows from table T1 in the data source db2. Every time Jane refreshes the view cache, both Jane and John would be able to view 20 rows, but every time John refreshes the view Jane can only view 10 rows.

- **Transaction Isolation levels** — Isolation levels specifying the degree to which one transaction must be isolated from data modifications made by other transactions. The following isolation levels are possible:
  - **Read Committed** — dirty reads are prevented; non-repeatable reads, and phantom reads can occur
  - **Read Uncommitted** — dirty reads, non-repeatable reads, and phantom reads can occur
  - **Repeatable Read** — dirty reads and non-repeatable reads are prevented; phantom reads can occur
  - **Serializable** — dirty reads, non-repeatable reads, and phantom reads are prevented

4. Click the **Advanced** tab to open the **Advanced** panel, and supply the following details:

- **Connection URL Pattern** is 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 Virtual View Manager does not validate modifications. The data source driver may or may not validate changes ignoring invalid URL connection parameters.

- **Connection URL String** is 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 driver to connect to the physical data source. This field is generated by the system and is un-editable.

For further details, see the “Connecting Through JDBC Drivers” topic in the “Post-Installation Tasks” chapter of the *IBM Cognos Virtual View Manager Administration Guide*.

- **JDBC Connection Properties** enables specification of Property - Value pairs passed to the targeted JDBC data source to set values that determine the data source behavior within the context of the connection with Virtual View Manager.

  Add custom JDBC Connection Properties for any JDBC data source. Currently a selection of commonly used properties for all the supported versions of MySQL, Oracle, and Sybase are pre-populated in the advanced connection properties window with default values. Other properties and values may be added as may be necessary with just a click on the expander button. Validation of correct property names is not provided by Virtual View Manager. The data source driver may simply ignore incorrectly named properties or invalid values or it may provide an error code.

- **Connection Pool Minimum Size** is the number of connections that should remain in the connection pool even when the pool becomes inactive.

  The connection pool is initially empty. When there is a need to connect to the data source, the pool creates one connection based on the information
connections remain available even if they become idle. If the maximum number of connections are already in use when a request comes in (even with regard to pass-through authentication), the new request is blocked and queued until the next connection is available, or it must wait until the Connection Pool Idle Timeout is reached.

If no connection was made available within the specified timeout, then a check is made for an available connection using the proper user and uses that, or if that is not available then the least recently used connection for some other user is dropped and a new connection for the required user is opened.

• **Connection Pool Idle Timeout** is the maximum waiting time (specified in milliseconds) for a new connection. The default timeout value is equivalent to 30 seconds.

• **Connection Validation Query** is a native data source query that is sent directly to the data source without evaluation by the Virtual View Manager query engine. Enter a simple query that will give a quick return. If the validation query returns a non-error result, then it will validate the connection to the data source.

Every time that a connection is checked out from the pool a connection validation query is executed to verify the validity of the connection. If the validation query fails that connection is discarded and a new connection is checked out from the available pool. Note that the connection validation query is data source specific and should be a simple native query, like one of the following example statements:

```
SELECT 1 FROM A_TABLE
SELECT 5 FROM A_TABLE WHERE 1 = 2
SELECT COUNT(*) FROM A_SYSTEM_TABLE
```

• **Execute SELECTs Independently** — If this option is selected, a SELECT statement submitted to this data source will be executed using a new connection from the connection pool and committed immediately after the SELECT is completed.

INSERT, UPDATE, and DELETE statements will continue to be executed using the same connection as part of the transaction.

5. Click **Next** to continue.
The data source with its resources is displayed in a tree format.

6. Do not yet expand the <data source name> node on the left. Notice the following fields and buttons on the panel:

- **Detect New Resources During Re-Introspection** — This option is available for every parent node you select in the tree display on the left (that is, for a data source, catalog, schema, or folder). The choice made here is used as follows during re-introspection:
  - If you select this option, child resources added to the current resource subsequent to this introspection will be detected during re-introspection. Suppose that the current resource is a catalog and a schema is added to this catalog subsequent to this introspection, that schema will be detected during re-introspection. For details on re-introspection, see “Data source re-introspection” on page 57.
  - If you do not select this option, child resources added to the current resource subsequent to this introspection will not be detected during re-introspection. Suppose the current resource is a catalog and a schema is added to this catalog subsequent to this introspection, that schema will not be detected during re-introspection. For details on re-introspection, see “Data source re-introspection” on page 57.

- The following fields are for supplying wildcard symbols to filter the names of catalogs, schemas, tables, procedures, and folders. The default symbol is given, but you can choose to supply a different symbol.
  - **Wildcard Symbol for Single Character Match** — Symbol that stands for a single character. The default symbol is the underscore (_). For example, if you type ab_ in the Schema Name Filter(s) field, it would match 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 (%). For example, if you type ab% in the Schema Name Filter(s) field, it would match schema-names such as abc, abd, ab, abaaaaa, 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 the backward slash (\). For example, the sequence ab\_c would match the resource-name ab\_c escaping the underscore (_).
  - **Separator for Each Filter** — Symbol for separating each filter. The default separator is a comma (,). For example, if you type orders,customers in the Table Name Filter(s) field, it would match the table names orders and customers.
  - **Schema Name Filter(s)** — Filters to select schemas. This field is displayed if the data source has schemas. See the preceding descriptions for wildcard symbols, escape character for wildcard symbols, and separator for filters.
  - **Catalog Name Filter(s)** — Filters to select catalogs. This field is displayed if the data source has catalogs. See the preceding descriptions for wildcard symbols, escape character for wildcard symbols, and separator for filters.
  - **Filter in Case-Sensitive Mode** — Makes the filter pattern to be case-sensitive. This option is not available for certain data sources.
such cases, the underlying physical data source determines the case-sensitive mode and filters the names of its resources as well for introspection.

7. Expand the `<data source name>` node, and notice all the schemas displayed.

8. Do one of the following:
   - Enter the value for filtering the schemas in the Schema Name Filter(s) field. For example, you can type SYS% if you want to select the schemas SYSCAT, SYSIBM, and SYSSTAT.
   - Or, select the desired schema in the tree displayed on the left.

9. Without expanding the schema node, enter the filters for tables and procedures.

10. Select the Detect New Resource During Re-Introspection box if you want the server to detect new resources that might be added to this schema later on.

11. Click Finish to complete the introspection.
   In the resource tree, you can see this data source displayed at the location chosen in step 1.

---

**Process to add an IBM Cognos Virtual View Manager Data Source**

Provided you have the right set of access privileges, you can introspect any database from IBM Cognos Virtual View Manager Data Services > Databases, and add it to the server as a data source.

The process of adding a data source is similar to adding a IBM DB2 data source with the following modifications:

- Select Virtual View Manager as the data source driver in the data source wizard and supply connection details that are specific to this data source.
- Note that the default port for the data source is 9401.
- The Authentication Domain is the domain where the Virtual View Manager server is installed.
- For details of introspection, see "Adding an IBM DB2 data source" on page 30.

**Adding DataDirect Mainframe data sources**

IBM Cognos Virtual View Manager connects to the Shadow RTE, which provides an interface to DataDirect Mainframe data sources.

The Shadow Interface for IBM DB2 provides integration with the DataDirect Mainframe IBM DB2 databases with features that provide: full relational metadata support, dynamic to static SQL conversion, full XA two phase commit, transaction and XML support, XML support, and event publishing.

**Before you begin**

The Shadow RTE Server (version 6.1.4.7606 or higher) must be installed on the DataDirect Mainframe computer and the Shadow RTE Client (version 6.1.1.1080 or higher) must be installed locally on the computer hosting the Virtual View Manager server.

The Virtual View Manager server must have the Shadow JDBC adapter driver (scjd12.jar). The Shadow JDBC adapter driver is available in the DataDirect Shadow Client installation directory:
Copy the scjd12.jar file into the following Virtual View Manager directory:
installation_location/apps/dlm/cis_ds_datadirect_mainframe/lib

When Virtual View Manager is running on a Windows operating system, Windows Services requires a system restart to recognize the new driver.

Connection to the Data Direct Mainframe is mediated by the Shadow RTE. Create a new Virtual View Manager server physical data source that points to the Shadow RTE of the DataDirect Mainframe.

The connection process parallels configuration of an IBM DB2 data source with some differences.

**Procedure**

1. Right-click in any personal or shared container within the Studio resource tree, and select **New Data Source**.
2. In the **Add Physical Data Source** window, select **DataDirect Mainframe**, and press **Next**. The **New Physical Data Source** window displays to allow connection configuration.

   All fields of the New Physical Data Source window must be filled to properly connect with the DataDirect Shadow RTE. Adding a DataDirect Mainframe data source to Virtual View Manager is similar to adding and configuring a connection with an IBM DB2 data source ("Adding an IBM DB2 data source" on page 30) with the following differences.

3. On the **Basic** tab of the **New Physical Data Source** window enter connection information and other relevant configuration details in the following fields:
   - **Name** — User-supplied name to represent the underlying data source. When the process of adding this data source is complete, the user-supplied name will be displayed in the resource tree.
   - **Host** — Name of the mainframe machine hosting the Shadow RTE or the host machine's IP address.
   - **Port** — Default port for the Shadow RTE is 1200, but that may be changed depending on implementation.
   - **DBMS Type** — DataDirect SQL access keyword that specifies the means by which the driver access the data.

   This keyword provides SQL access to databases by specifying the means by which users will be accessing data. IBM DB2 is the storage scheme that allows for the most efficient push down of SQL to the DataDirect Mainframe. Other DBMS types don’t allow as much SQL push down. Most of the storage schemes listed below will not allow the Virtual View Manager to push SQL optimizations to the mainframe. Supported DBMS type values are as follows:
   - **ADABAS** — very fast transaction processing mainframe database, driver is not optimized for pushing SQL to the source.
   - **DATA** — may be utilized to access both VSAM and sequential files, not optimized for SQL push to the source.
   - **DB2** — provides the broadest coverage of SQL-99 functionality, this setting requires specification of a database name.
   - **IMSDB** — used for access to multi-dimensional databases, the results from queries sent using IMSDB may appear odd compared to the results returned from a conventional set of relational tables.
- **VSAM** — this storage system that may be used for read-only
- **VSAM/CICS** — transaction processing data system for online and batch systems that allows both read and write access.

Each physical data source connection may utilize only one DBMS type. If the DataDirect Mainframe utilizes more than one DBMS type then more data source connections may be created in Virtual View Manager to access data stored using those different schemas.

- **Database** is equivalent to **Database Name** in the IBM DB2 configuration — Name of the underlying data source. Entries in this field are used only for IBM DB2 DBMS types. For other DBMS types enter none.

- **Plan** specifies the data isolation plan. The default value, SDBC1010, specifies cursor stability. Other values may be specified to opt for Repeatable reads, Read Stability, or Uncommitted reads. Refer to the Shadow RTE Client/Drivers Installation and Administration Guide for more information.

- **Login** and **Password** — Valid user name and password for the Shadow RTE/DataDirect Mainframe.

- **Save Password** — This option works in combination with the **Pass-through Login** option. By default, this option is enabled and un-editable. It becomes editable when you select the **Pass-through Login** option.

  If you accept the default, the password is saved and the Pass-through Login option remains disabled. When the password is saved the following operations may be performed without having to supply the password again:
  - Introspect the current data source
  - Re-introspect the data source
  - Add/Remove data source resources
  - Perform query/update/insert operations on a table in the data source (provided appropriate privileges are granted on the resource)
  - Invoke a stored procedure (provided appropriate privileges)
  - Refresh a cached view based on the data source resources
  - Use the query optimizer for statistics gathering

- **Pass-through Login** — Works in combination with the Save Password option.

  If you accept the default, this mode stays Disabled and the password is saved. Such mode here is referred to as the non-pass-through mode. Refer to the details given for the Save Password option.

  If you select the Enabled option, the Save Password option becomes editable. Such mode here is referred to as the pass-through mode.
  - In the pass-through mode, if you save the password, you can introspect without resupplying the password.
  - In the pass-through mode, if you don’t save the password, you can perform the following operations:
    - Query, update, insert and delete operations. You need to re-supply the original login credentials for the current session.
    - Re-introspection, Add/Remove data source resources. You will be prompted to re-supply the same password that was used when the data source was originally introspected.
  - In the pass-through mode, if you don’t save the password, you cannot perform the following operations:
    - Schedule re-introspection
    - Statistics gathering, using the query optimizer
Note: If a view uses a data source that was originally added to Virtual View Manager Server using the pass-through mode without saving the password, row-based security may affect the cache refresh functionality. For example, suppose that a cached view, named CachedCommonView, uses the SQL statement `SELECT * FROM db2.T1` and the user John with the password john1 is allowed only to view 10 rows, whereas the user Jane with the password jane1 is allowed to scan 20 rows from table T1 in the data source db2. Every time Jane refreshes the view cache, both Jane and John would be able to view 20 rows, but every time John refreshes the view Jane can only view 10 rows.

- **Transaction Isolation** levels — Isolation levels specifying the degree to which one transaction must be isolated from data modifications made by other transactions. The following isolation levels are possible:
  - **Read Committed** — dirty reads are prevented; non-repeatable reads, and phantom reads can occur
  - **Read Uncommitted** — dirty reads, non-repeatable reads, and phantom reads can occur
  - **Repeatable Read** — dirty reads and non-repeatable reads are prevented; phantom reads can occur
  - **Serializable** — dirty reads, non-repeatable reads, and phantom reads are prevented

The Advanced tab on the New Physical Data Source window is exactly analogous to the IBM DB2 configuration window aside from the Connection URL Pattern which has Shadow RTE specific arguments. Refer to the IBM DB2 description of the Advanced tab for more info on the Connection Pool fields.

- **Connection URL Pattern** contains the following keywords which are set with the Basic tab field entries.
  - APNA sets the application name for SQL grouping.
  - CPFX sets the catalog prefix for IBM DB2 or non-IBM DB2 resources.
  - DBTY sets the DBMS type.
  - HOST and PORT set the network name or IP address and the port number.
  - SUBSYS is used for IBM DB2 database names.
  - TRLT turns off truncation for strings greater than 20 characters.
  - DTFM sets the date format to standard ODBC/ISO format: yyyy-mm-dd

Refer to Appendix A of the *Shadow RTE Client/Drivers Installation and Administration Guide* for more information on the Shadow driver keywords.

### IBM Informix data sources

You should manually install the JDBC driver that would connect your IBM Informix data source to IBM Cognos Virtual View Manager.

For details on obtaining the driver, refer to "Installing and Using Special JDBC Drivers" in the chapter "Post-Installation Tasks" in the *IBM Cognos Virtual View Manager Administration Guide*.

Adding an IBM Informix data source to Virtual View Manager is similar to introspecting a IBM DB2 data source with the following modifications:

- Select IBM Informix 9.x as the data source driver in the data source wizard and supply connection details that are specific to this data source.
Note that the default port is 1526.

- **Server** is the name of the IBM Informix instance.

Also, note that IBM Informix stored procedure parameters are not introspected properly. Therefore, you need to manually define the parameters by using the Design Mode in the Parameters panel. See “About editing stored procedures” on page 117.

For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

---

**Microsoft SQL Server data sources**

You should manually install the JDBC driver that will connect your Microsoft SQL Server data source to the IBM Cognos Virtual View Manager Server.

For details on obtaining the driver, refer to the “Installing and Using Special JDBC Drivers” topic in the “Post-Installation Tasks” chapter of the *IBM Cognos Virtual View Manager Administration Guide*.

Adding a Microsoft SQL Server data source is similar to introspecting a IBM DB2 data source with the following modifications:

- Select Microsoft SQL Server as the data source driver in the data source wizard and supply connection details that are specific to this data source. Note that the default port is 1433.

- The **Advanced** properties section offers properties named **Select Mode** and **Show All Databases**.

  - **Show All Databases** lists all the available databases during introspection. For example, while providing the details for connecting to a specific database using the Microsoft SQL Server driver, if you specify the desired database to be Northwind and also select the option **Show All Databases**, you will be able to view all the databases including Northwind that exist in the system. So, you can choose resources from several databases.

  - **Select Mode** has two values: **Cursor** and **Direct**. These two values affect how result sets are created and retrieved when a query is executed against a Microsoft SQL Server data source.

    The direct method sends the complete result set in one request to the driver. It is useful for queries that only produce a small amount of data that you fetch completely. You should avoid using direct when executing queries that produce a large amount of data, as the result set is cached completely on the client and constrains memory. In this mode, each statement requires its own connection to the database. This is accomplished by cloning connections. Cloned connections use the same connection properties as the original connection; however, because transactions must occur on a single connection, auto commit mode is required. Due to this, JTA is not supported in direct mode. In addition, some operations, such as updating an insensitive result set, are not supported in direct mode because the driver must create a second statement internally. Exceptions generated due to the creation of cloned statements usually return an error message similar to “Cannot start a cloned connection while in manual transaction mode.”

    When the SelectMethod is set to cursor, a server-side cursor is generated. The rows are fetched from the server in blocks. The JDBC Statement method **setFetchSize** can be used to control the number of rows that are fetched per...
request. The cursor method is useful for queries that produce a large amount of data, data that is too large to cache on the client.

MySQL data sources

Adding a MySQL data source is similar to introspecting a IBM DB2 data source, with some modifications.

The modifications are:

- Select MySQL as the data source driver in the data source wizard and supply connection details that are specific to this data source. Note that the port number default is 9408 or the base port setting plus eight, depending upon the MySQL instance.
- The Advanced properties section offers a property named Streaming Results Mode. If selected, this option sends the result set from MySQL to IBM Cognos Virtual View Manager in streams. It gives a signal to the driver to stream the result set row-by-row. If this option is not selected, MySQL will not send any results until all the results are gathered. For further details on this mode, refer to the README.txt file in MySQL JDBC driver’s docs directory.
- Connection Validation Query — MySQL data sources may use a statement like “SELECT 1” as a validation query. The connection validation query is a native query that does not pass through the Virtual View Manager query engine but instead gets submitted directly to the MySQL data source to validate the connection.

For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

Netezza data sources

You should manually install the JDBC driver that would connect your Netezza data source.

For details on obtaining the driver, refer to the "Installing and Using Special JDBC Drivers" topic in the "Post-Installation Tasks" chapter of the IBM Cognos Virtual View Manager Administration Guide.

Adding a Netezza data source is similar to introspecting a IBM DB2 data source with the following modifications:

- Select Netezza as the data source driver in the data source wizard and supply connection details that are specific to this data source. Note that the default port is either 5480 or the default, depending upon the Netezza instance.
- For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

Oracle data sources

IBM Cognos Virtual View Manager supports Oracle data sources.

Adding an Oracle Thin Driver Data Source

Adding an Oracle data source is similar to introspecting a IBM DB2 data source, with some modifications.

The modifications are:
Select Oracle as the data source driver in the data source wizard and supply connection details that are specific to this data source. Note that the default port is different (1521).

Service Name is the name of the Oracle database service that specifies the database to be used.

For other details of introspection, see "Adding an IBM DB2 data source" on page 30.

**Oracle OCI driver data sources**

You need to use an Oracle client to create the data source. Information about the types of Oracle clients supported is given here.

**Oracle 9i client**

Oracle 9i client lets users to connect to Oracle 9i Oracle 10g databases. You should have Oracle 9i client installed on the machine that is hosting IBM Cognos Virtual View Manager. Using Oracle 9i client, you can add Oracle 9i (OCI Driver) and Oracle 10g (OCI Driver) data sources to the Virtual View Manager server. You must also use the appropriate Oracle JDBC JAR. For details, see "Setting up the environment for using an Oracle OCI driver."

**Oracle 10g client**

Oracle 10g client lets users to connect only to Oracle 9i and Oracle 10g databases. Using Oracle 10g client, you can add Oracle 9i (OCI Driver) and Oracle 10g (OCI Driver) data sources.

**Setting up the environment for using an Oracle OCI driver**

Follow these steps to set up your environment to use an Oracle OCI driver.

**Procedure**

1. Install an Oracle client instance on the machine where Virtual View Manager is running.
   
   This Oracle client instance must have Oracle net services components for the OCI driver. Two Oracle client types, namely Oracle Administrator and Runtime clients, contain net services components by default. Refer to Oracle documentation about how to set up an Oracle client.

2. Set up TNS names.
   
   TNS names are created in Oracle Net Configuration Assistant. These TNS names contain information about how to connect to a physical Oracle database. Refer to Oracle documentation for details on how to set up TNS names.

3. Set up environment variables on the machine that hosts Virtual View Manager.
   
   On Windows:
   
   - Set the ORACLE_HOME environment variable.
   - Add ORACLE_HOME\lib to System PATH.
   - Set the CLASSPATH environment variable to use Oracle JDBC driver from ORACLE_HOME\jdbc\lib.

   On Linux or UNIX:
   
   - Set the ORACLE_HOME environment variable.
   - Add ORACLE_HOME/lib to LD_LIBRARY_PATH environment variable.
• Set the CLASSPATH environment variable to use Oracle JDBC driver from ORACLE_HOME/jdbc/lib.

4. Stop the server.

5. Make sure that Virtual View Manager is using the same JDBC driver as the Oracle client.

6. Restart the server.
   Now you can add Oracle data source types that use the OCI driver, as described next.

Adding an Oracle OCI data source
Follow these steps to add an Oracle OCI data source.

Procedure
1. Set up the environment for using an Oracle OCI driver (see “Setting up the environment for using an Oracle OCI driver” on page 41).
2. Right click on the resource tree and select New Data Source.
3. Select the data source, and click Next.
4. Supply the following connection details in the Add Physical Data Source panel:
   • Name — the data source name in Virtual View Manager.
   • Service Name — the TNS name (Net Service Name) that is set up through Oracle Net Configuration Assistant.
   • Login — the login name to access Oracle database.
   • Password — the password to access Oracle database.
5. Open the Advanced Properties tab and scroll the window to the bottom.
   To include Oracle views that are labeled with a status of invalid select the check box named Include Invalid Introspection Objects. Otherwise the introspection of the Oracle data source will exclude views with a status of invalid even though the Oracle invalid state may be simply a lack of current validation.
   For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

Sybase data sources

Adding a Sybase data source is similar to introspecting a IBM DB2 data source, with some modifications.

The modifications are:
• Select Sybase as the data source driver.
   The default port is 4100, but local implementation may be any port setting.
• The Advanced properties section offers a property named Show All Databases which lists all the databases that could be connected to using the Sybase driver.
   See also the details given in the section “Microsoft SQL Server data sources” on page 39.
• Ignore Procedure Return Parameter — To suppress or ignore the success/failure return parameter for Sybase (and SQL) stored procedures mark the ignore procedure return parameter check box when JDBC clients do not utilize or wish
to view the procedure return parameter. Otherwise a return parameter will be
inserted into procedure definitions by the JDBC driver.

- **Introspect Should Use Column Alias** — Sybase data sources may display and
  use column names (default setting) or column aliases. If column aliases are
  desired then use the Advanced tab when defining the data source, and check the
  **Introspect Should Use Column Alias** check box.

- **Connection Validation Query** — Sybase data sources may use a statement like
  “SELECT 1” as a validation query. The connection validation query is a native
  query that does not pass through the Virtual View Manager query engine but
  instead gets submitted directly to the Sybase data source.

For other details of introspection, see [“Adding an IBM DB2 data source” on page 30](#).

---

**Teradata data sources**

Using the Teradata JDBC driver, IBM Cognos Virtual View Manager may introspect
Teradata V2R6.0 and V2R6.1 data sources.

Earlier releases may also function, but they are not supported.

**Teradata JDBC driver installation**

The Teradata JDBC driver jars are available from the Teradata Web site.

Add the driver files named tdgssconfig.jar, tdgssjava.jar, and terajdbc4.jar to the
<installation directory>\apps\dlm\cis_ds_teradata\lib directory where Virtual View
Manager is installed.

Virtual View Manager is already configured to use the drivers, but a restart of
server will be necessary to initiate use of a new driver jars.

**About adding Teradata data sources**

Adding a Teradata data source is similar to introspecting a IBM DB2 data source,
with some differences.

The differences are:

- When selecting Teradata as the data source driver, the default port is 7060.
- The Teradata connections URL pattern takes the following form:
  
  \jdbc:teradata://<HOST>/DBS_PORT=<PORT>/DATABASE=<DATABASE_NAME>/
  CHARSET=UTF8,COMPAT_DBS=true

- Check the **Show All Databases** check box in the Advanced properties tab to
  introspect the source and display a list of all the available databases and tables
  accessible with the Teradata driver. If the check box is not selected, then only the
  database specified to the given user Login/Password pair will be shown after
  introspection.

- Introspection of a Teradata RDBMS data sources reveals tables, views,
  procedures, functions, and macros.

  Introspected Teradata functions and macros appear as procedures in the
  associated Virtual View Manager display of available Teradata resources.
  Teradata macro parameters map to Virtual View Manager parameters with the
  appropriate Virtual View Manager data types and references for input of values
  and return of results.
• Teradata data type mappings to Virtual View Manager JDBC data types are documented in Chapter 5 of the IBM Cognos Virtual View Manager Reference Guide in the section on Teradata Data Types and JDBC Data Types.
• To view debug messages the Connection URL Pattern of the Teradata data source may be modified from the Advanced tab of the Data Source configuration window by appending the parameter 
  \,LOG=DEBUG
• The Teradata driver does not send debug log messages to a file, but instead they are sent to system.out.

**Microsoft Access data sources**

For adding a Microsoft Access data source, you must know its DSN (Data Source Name) which was defined when the data source was added to the host machine.

**Adding a Microsoft Access data source**

Follow these steps to add a Microsoft Access data source.

**Procedure**

1. Right-click within the resource tree, and select **New Data Source**. In the **Add Physical Data Source** window, select **Microsoft Access** for the driver.
2. In the window for providing data source details, supply the relevant information in the following fields:
   • **Name**—User-defined name for the data source
     When the process of adding this data source is complete, the user-defined name will be displayed in the resource tree representing the underlying data source.
   • **Login and Password**—Valid user name and password, if any, to access the underlying data source.
   • **Save Password and Pass-through Login**—See under "Adding an IBM DB2 data source" on page 30.
   • **DSN**—DSN (data source name) for the data source. A new User or System DSN may have to be created using the ODBC Data Source Administrator utility. The ODBC Data Source Administrator utility is available with all supported Windows operating systems.
     This is the DSN (data source name) you provided when you added the data source to your computer. Virtual View Manager uses this name to find and connect to the underlying data source.
     An ODBC User data source (User DSN) stores information about how to connect to the indicated data provider/file. A User data source is only visible to the current user and can only be used on the current machine.
     An ODBC System data source (System DSN) stores information about how to connect to the indicated data provider. A System data source is visible to all users on the current machine, including NT services.
     An ODBC File data source (File DSN) lets you connect to a data provider. File DSNs can be shared by users who have the same driver installed.
   • **Character Set**—See "Characteristics of different file formats" on page 52.
3. In the **Add Physical Data Source** screen, click **Next** to continue.
4. Expand the **data source name** node and other relevant nodes, select the tables you want to use for your modeling, and click **Finish**.
Microsoft Excel data sources

Adding a Microsoft Excel data source is similar to adding a Microsoft Access data source, except that you must select Microsoft Excel to be the data source driver. Also, you don’t need a user name and password to introspect an Excel data source.

See also “Microsoft Access data sources” on page 44.

LDAP data sources

IBM Cognos Virtual View Manager can introspect and add an LDAP data source and make it look as if it were a relational table.

Adding LDAP data sources

Follow these steps to add an LDAP data source.

Procedure

1. Select LDAP as the data source driver.
2. Type the path to the LDAP data source in the URL field, in the following format:

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

   Note that the directory suffix (as the oin o=<organization_name>) depends upon how the LDAP is set up.
   In most cases, the LDAP may not need a user name and password.
   For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

WSDL data sources

This section introduces WSDL and web services, and describes how to add a WSDL SOAP 1.1 compliant data source.

We also support SOAP over JMS if the WSDL is described using either TIBCO or the Apache WSIF specification.

WSDL and web services

Web Services Definition Language (WSDL) is the language that describes XML documents passed across web services. Web services are web-based applications that dynamically interact with other web applications using an XML message protocol such as SOAP 1.1.

One of the standards for describing these services is WSDL 1.1. Web services can be bound to any message format and protocol, but three bindings are popular: SOAP, HTTP, and MIME. Virtual View Manager 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 defines what mechanism is used for transporting the request.
- Virtual View Manager supports HTTP and JMS transport.

The encoding scheme defines how the message is encoded for the transport.
• Virtual View Manager supports the literal encoding scheme, which uses an XML schema as the definition for how data should be encoded.
• Virtual View Manager also provides a limited support for the SOAP encoding scheme as defined in the SOAP 1.1 specification.
• Virtual View Manager does not support multi-dimensional or sparse SOAP arrays.

The message style refers to how the request itself is structured.
• Virtual View Manager supports the Document Literal and RPC message styles.
• The Document Literal style messages have a single part whose schema defines the message payload.
• The RPC style messages have a payload that is defined by a combination of rules that are defined in the SOAP 1.1 specification and the schema that defines individual parts.

About introspecting WSDL data sources
WSDL data sources are a class of web services. Introspecting a WSDL data source is similar to introspecting a IBM DB2 data source, with some modifications.

The modifications are:
• Select WSDL as the data source driver.
• In the Basic panel, supply the path to the WSDL data source in the URL field. WSDLs may be available by URL over any accessible network. A locally mapped WSDL can be introspected using a URL format such as file:///Z:/test.wsdl
In most cases, the WSDL source does not need a user name and password.
• The Advanced Panel enables import of trusted certificates after completing the introspection of the data source.

After clicking Next, the WSDL specified is introspected and the selected services, ports, and operations are made available for view. Expand the nodes of the web service and ports to see the operations within each service, or if you know you want all the data services either check the box at the top level, or click Select All.
• Introspection properties are available for JMS and HTTP ports. For JMS ports:
  – The Connector should be specified to ensure proper functionality of the JMS data source connection pools. JMS connectors should be installed by an administrator. Refer to the IBM Cognos Virtual View Manager Administration Guide.
  – The specified JMS Destination may be changed to take advantage of different queue destination aliases that offer the same service.
  – Delivery Mode may 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 acknowledgement 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 will specify that the port setting will take precedence.
  – Operations or messaging priority may be set an integer within the range of 1 to 9. Where 9 is the highest priority.
  – Default Timeout is a setting for the consuming client and it may be set to some duration measured in milliseconds, where an entry of zero means no timeout, and a null entry will specify that the default will take precedence.
Individual JMS operations under the port can be configured with a Message Type of Bytes or Text and with specific timeouts tailored to that particular process.

For HTTP ports and operations only the timeout parameter in milliseconds may be specified, where an entry of zero means no timeout, and a null entry for an operation will specify that the port setting will take precedence.

If configurations for the web service need to be reviewed or changed at any time, simply open the web service from Virtual View Manager and click the Add/Remove Resources, make required changes, and re-introspect the data source.

For other details of introspection, see “Adding an IBM DB2 data source” on page 30.

Representation of a WSDL data source in the resource tree:

A WSDL data source is added to the resource tree in four hierarchical nodes represented with the wing icons:

- WSDL Data Source name
- Web Service
- Web Service Port
- Web Service Operation

If the data source contains a service or operation that Virtual View Manager does not support, the resource tree does not display that service or operation. For every object that is not supported, an error is generated in the log file (cs_server.log).

Using a WSDL data source:

You can use a WSDL data source in many ways.

You can directly execute the WSDL operation:

Double-click the WSDL operation in the resource tree to open its editor.

Click Execute toolbar ( ) in the editor.

To publish a WSDL data source, you should first transform the WSDL data into a transformation, and then publish the transformation. For details on transformations, see “Transformations” on page 100.

Virtual View Manager also offers a mechanism for specifying how the web service operation request message parts and response message parts should be mapped respectively to input and output parameters.

To map WSDL request message parts and response message parts to input and output parameters

Open the WSDL data source port or the operation. Settings made on the port are propagated by default to the operations if the Use settings from parent Web Service Port check box is enabled.

Right-click on the port/operation and select Open.
In the editor that opens on the right, click the Parameter Mapping tab.

The Parameter Mapping panel displays options for specifying operations request message parts and response message parts for mapping to input and output parameters.

To view the request and response messages, you can create a message pipeline step called Log Message to File and view the result after executing the operation.

For details on pipeline steps, see "Message level security" on page 169.

### Adding 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.

**Procedure**

1. Create an XML-type definition set.
   
   This definition set provides the output document definition for retrieving XML data over HTTP.
   
   Definition sets have been introduced in the first chapter of the Getting Started Guide. Quick steps for creating an XML definition set are given in this section. For more details, see Chapter 6, “Definition sets,” on page 127.

2. Create an XML/HTTP data source using the XML-type definition set. See these sections for how to do this:
   - "Creating XML/HTTP data sources - 1 (using GET without sending a request)" on page 49,
   - "Creating XML/HTTP data sources - 2 (using GET and sending a request)" on page 50
   - "Creating XML/HTTP data sources - 3 (using POST)" on page 50

**Results**

After adding the XML/HTTP data source, if you want to have the XML data in tabular form in order to use it in SQL queries, you should transform the XML data. For details on transformation, see "XML data retrieval over HTTP" on page 51.

### Creating XML-type definition sets

This section gives quick steps for creating an XML-type definition set.

For further details on definition sets, see Chapter 6, “Definition sets,” on page 127.

**Procedure**

1. Right-click at an appropriate location in the resource tree, and select **New Definition Set**.

2. In the input dialog:
   - Type a name for the definition set.
   - Select **XML** in the **Type** drop-down list, and click **OK**.
     
     The definition set is added to the resource tree, and the editor for the definition set opens in the right pane. Now you're ready to design the definition set.
3. In the editor, select the XML Schema Text tab in order to define your XML schema. You can either type your XML schema in this panel or upload an existing XML schema by using the Insert From File toolbar button in the editor.

4. Once you have the desired XML schema in the XML Schema Text panel, save it. You can also edit the schema as necessary.

5. Use the Parse XML Schema toolbar button to parse the schema after editing the schema. Parsing the schema creates XML element declarations and type definitions from the schema, which you can find in the XML Schema panel.

GET/POST method to add XML/HTTP data sources

This section describes how to connect to XML/HTTP data sources using GET and POST methods and add them to Virtual View Manager.

Creating XML/HTTP data sources - 1 (using GET without sending a request)
The example used here illustrates the GET method and does not send any input to the HTTP server.

Procedure

1. Right-click at an appropriate location in the resource tree, and select New Data Source.

2. In the Add Physical Data Source wizard, select XML/HTTP for the data source driver. The window for providing the following connection details opens (see the screen at the end of this procedure description).
   - Name—user-supplied name for the 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 in order 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 and Password—valid user name and password to access the HTTP server.

   - Save Password and Pass-through Login—See “Adding an IBM DB2 data source” on page 30.

   - No Input—option to indicate that you are not supplying any input document definition or an input in the query string

   To specify the output document definition, use the Browse button to locate the definition set you created as described in “Creating XML-type definition sets” on page 48. When you click the Browse button, the Choose Schema window displays.

   In the left pane, select the XML definition set in the resource tree.

   In the right pane, make sure that Element is selected in the Show drop-down list, and select the definition (rss in the screen).
When the 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 then the document element must have the same type as the type selected.

When you make your selection, the Schema field is automatically filled with the complete path to the output document definition, as shown in the next screen.

- Click OK.

You return to the Add Physical Data Source window.

- Now in the Add Physical Data Source window, notice that the Output Document Definition field is filled with the same information as that of the Schema field in the previous screen.

- Click Finish to complete the process of adding the data source.

**Creating XML/HTTP data sources - 2 (using GET and sending a request)**

The example used here also illustrates the GET method but sends a request to the HTTP server.

**Procedure**

1. Follow step 1 in the section “Creating XML/HTTP data sources - 1 (using GET without sending a request)” on page 49, and read through the descriptions in step 2.

2. Supply the Basic connection details as follows:
   - Type the path to the HTTP server in the URL field.
   - Select GET in the Method list box.
   - Select Input In Query String, and type the input in the URL field as in the following example:
     ```
     http://<server_name>/cgi-bin/
     GetVotersByGenderAndParty1.cgi?gender={PARAM gender}
     &party={PARAM party}&limit={PARAM limit}
     ```
   - Select Output Document Definition, and type the path to the output document definition, which would supply the schema for retrieving XML data, as in the following example:
     ```
     /<path to definition set>/xmlhttp_ds_schema/GetVotersResponse
     ```
   - Or, use the Browse button to locate the path to the Output Document Definition.

   - Click OK to select the schema.

   You return to the Add Physical Data Source window.

3. Click Finish to complete the process of adding the data source.

**Creating XML/HTTP data sources - 3 (using POST)**

The example used here illustrates the POST method and sends an input document definition to the HTTP server.

**Procedure**

1. Follow step 1 in the section “Creating XML/HTTP data sources - 1 (using GET without sending a request)” on page 49, and read through the descriptions in step 2.

2. Enter the connection details as follows:
Enter the absolute path to the HTTP server in the URL field.
Select POST in the Method list box.
Select Input Document Definition, and type the input as in the following example:

```
/path/to/definition Set>/xmlDefinitionSetForVotersList/GetVotersRequest
```
You can also use the Browse button to locate this path.
Click OK to select the schema.

3. In the Add Physical Data Source window, click Finish to complete the process of adding the data source.

**XML data retrieval over HTTP**

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.

**Viewing data in XML format**

Follow these steps to view data in XML format.

**Procedure**

1. Double-click the XML source (also called Web service operation) within the desired 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 button.
   The result is displayed in the Result panel.
3. Click the Details button to view the XML data.

**Creating transformations of XML/HTTP data sources**

Follow these steps to create a transformation of an XML/HTTP data source.

**Procedure**

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.
3. In the tree display, select the XML source in the desired XML/HTTP data source for the transformation.
4. Type a name for the transformation in the Transformation Name field, and click Finish.
   For further details on transformations, see the Procedures chapter.

**Executing transformations to get XML data in tabular format**

Follow these steps to execute the transformation and get XML data in tabular format.
Procedure
1. Follow the steps described in “Mapping source XML data to target output columns” on page 102.
2. Execute the transformation, as described in “Executing transformations” on page 122.

File data sources

A file data source refers to the following files and features:
- It is a file with value separators or is an XML file.
- If the file does not have a header row, the column names are determined automatically.

You can use any valid system file.

Characteristics of different file formats

This section describes the characteristics of different file formats.

File with a delimiter
- The file may have a header with each entry separated by a comma, tab, or any other single character.
- The file can use alphabetical characters for delimiting.
- The file has a text qualifier (with default quotes) that allows the delimiter to be used in the data.
- The file can use any one of the character-encoding types listed in the following table.

Table 3. Character encoding set

<table>
<thead>
<tr>
<th>Name</th>
<th>Same As</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cp1250</td>
<td>windows-1250</td>
<td>Windows Eastern European</td>
</tr>
<tr>
<td>Cp1251</td>
<td>windows-1251</td>
<td>Windows Cyrillic</td>
</tr>
<tr>
<td>Cp1252</td>
<td>windows-1252</td>
<td>Windows Latin-1</td>
</tr>
<tr>
<td>Cp1253</td>
<td>windows-1253</td>
<td>Windows Greek</td>
</tr>
<tr>
<td>Cp1254</td>
<td>windows-1254</td>
<td>Windows Turkish</td>
</tr>
<tr>
<td>Cp1255</td>
<td>windows-1255</td>
<td>Windows Hebrew</td>
</tr>
<tr>
<td>Cp1256</td>
<td>windows-1256</td>
<td>Windows Arabic</td>
</tr>
<tr>
<td>Cp1257</td>
<td>windows-1257</td>
<td>Windows Baltic</td>
</tr>
<tr>
<td>iso-8859-1</td>
<td>iso-8859-1</td>
<td>iso-8859-1, Latin Alphabet No. 1</td>
</tr>
<tr>
<td>US-ASCII</td>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>UTF-8</td>
<td>UTF-8</td>
<td>Eight-bit UCS Transformation Format</td>
</tr>
<tr>
<td>UTF-16</td>
<td>UTF-16</td>
<td>Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark</td>
</tr>
<tr>
<td>UTF-16BE</td>
<td>UnicodeBigUnmarked</td>
<td>Sixteen-bit Unicode Transformation Format, big-endian byte order</td>
</tr>
<tr>
<td>UTF-16LE</td>
<td>UnicodeLittleUnmarked</td>
<td>Sixteen-bit Unicode Transformation Format, little-endian byte order</td>
</tr>
</tbody>
</table>
Table 3. Character encoding set (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Same As</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>windows-1250</td>
<td>Cp1250</td>
<td>Windows Eastern European</td>
</tr>
<tr>
<td>windows-1251</td>
<td>Cp1251</td>
<td>Windows Cyrillic</td>
</tr>
<tr>
<td>windows-1252</td>
<td>Cp1252</td>
<td>Windows Latin-1</td>
</tr>
<tr>
<td>windows-1253</td>
<td>Cp1253</td>
<td>Windows Greek</td>
</tr>
<tr>
<td>windows-1254</td>
<td>Cp1254</td>
<td>Windows Turkish</td>
</tr>
<tr>
<td>windows-1255</td>
<td>Cp1255</td>
<td>Windows Hebrew</td>
</tr>
<tr>
<td>windows-1256</td>
<td>Cp1256</td>
<td>Windows Arabic</td>
</tr>
<tr>
<td>windows-1257</td>
<td>Cp1257</td>
<td>Windows Baltic</td>
</tr>
</tbody>
</table>

XML file characteristics
- Can have a schema.

**Location of files**

The file (delimited or XML) you would like to use can reside either on a computer where the Virtual View Manager server is running or on a computer that is available to the Virtual View Manager server via a URL. The URL should be relative to the computer where the Virtual View Manager server is running.

If the computer on which the file is located does not have a Web server, it must be mapped to the computer where the Virtual View Manager server is running.

**Adding file-delimited data sources**

Follow these steps to add a file-delimited data source.

**Procedure**

1. Right-click at an appropriate location in the resource tree, and select **New Data Source**.
2. In the **Add Physical Data Source** window, select **File - Delimited** for the driver, and click **Next**.
3. Supply the relevant information in the fields, and click **Next**:
   - **Name**—User-defined name for the data source
     When the process of adding this data source is complete, the user-defined name will be displayed in the resource tree representing the underlying data source.
   - **Local File System**—If the file is on the local file system, select this option.
     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. And, 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 be exactly the same.
     See also "**Location of files**"
   - **Root Path**—Specifies where to begin the search for the desired file(s). Root path is the absolute path to the root directory where the desired files reside.
     You can create a file data source using the root path to query the files residing in the root directory. For example, you can create a file data source,
say myFileDataSource, at shared/sources to query myFile1.txt (which located in C:\files\FlatFilesDirectory). The SQL of such a query would be of the following form:

```
SELECT *
FROM /shared/sources/myFileDataSource/myFile1.txt
```

Note that the query is independent of the absolute path.

If this data source is exported from a staging machine and imported to a production machine, the path for the logs directory may 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, you must run re-introspection to ensure the existence of all the files. If the file structure of the new location is different from the old one, it will entail adding, deleting, or changing some of the files.

When the root directory is set, you can only see (or introspect) files under it. You can never introspect files beyond the root directory.

- **URL**—File is on a network. The URL to the file should be relative to the computer where the IBM Cognos Virtual View Manager server is running. With this option, you can select only one file. If the machine housing the file does not have a Web server, it must be mapped to the machine where the Virtual View Manager server is running.

- **Character Set**—Character encoding type. See “Characteristics of different file formats” on page 52.

- **Delimiter**—File delimiter. The options are comma (,), colon (:), semi-colon (;), period (.), forward slash (/), backward slash (\), and tab.

- **Text Qualifier**—Qualifier within which the whole text in the file is enclosed.

- **Has Header Row**—Select true, if the file text has a header row; otherwise, select false.

- **File Name Filters**—Lets you automatically restrict what types of files are to be introspected. For example, if you want to introspect only files with the extension .csv, you can specify *.csv as the filter. Then during introspection, the introspection tree will only display the file names with this extension. If you want to introspect more than one type of files, your filter for each type should be separated by a comma (,).

Example of a single filter: *.csv

Example of two filters: *.csv,*.txt (a space is allowed between two filters)

Rules for the filters:

a) An asterisk (*) means that any character in the file name occurs zero or more times.

b) A question mark (?) means that any character in the file name occurs exactly once.

c) A comma (,) is a separator for each filter.

d) A backward slash (\) is an escape character to escape a file name that contains an asterisk, question mark, or a comma.

Only files that match the filter you specify here will be exposed later on during the process of adding/removing data source resources (through the **Add/Remove Resources** menu option) and also during data source re-introspection.

4. Expand the <data source> node, select the desired folders and/or files.
5. Select **Detect New Resources During Re-Introspection**, if you want new files of the same type to be detected later on in the re-introspection process.

6. Click **Finish** to complete the process of adding the data source.

---

### File-cache data sources

Adding a file-cache data source is similar to adding a delimited file data source.

For more information, see [“Adding file-delimited data sources” on page 53](#). The file-cache data source is used for storing resource data that are cached using the Automatic storage mode. For additional information, see [Chapter 11, “Caching resources,” on page 193](#). 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.

### File-XML data sources

Adding an XML file is similar to adding a file with delimiters except that you should select **File - XML** for the driver, and provide a schema location for the XML data source.

You can however, let the system decide the XML schema for you by leaving the **Schema Location** field blank. 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.

The syntax for the location is

```
<namespace> <location> [ <namespace> <location> ]
```

where `<namespace>` is the target namespace for the XML schema and `<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.

The **No Name Space Schema Location** property is similar to the **Schema Location** property in that it is used to locate a schema that matches the selected instance documents. However, unlike the **Schema Location** property, the **No Name Space Schema Location** property contains only a single URL which is the location of a schema document that does not define a target namespace.

### How to add an XML Data Source

The procedure to add an XML data source is similar to adding a WSDL data source.

However, note that unlike in the case of a WSDL data source, an XML file data source cannot be executed before being transformed. For more information, see [“WSDL data sources” on page 45](#).

### Custom Java procedures

Virtual View Manager supports custom procedures written in Java to query data sources. APIs are provided to create custom procedures.

For details on Virtual View Manager APIs to create custom Java procedures, see the chapter “Java APIs for Custom Procedures” in the *IBM Cognos Virtual View Manager Reference Guide.*
You can add custom Java procedures to Virtual View Manager just as you would add any other type of data source.

**Preparing custom Java procedures**

Before you add a custom Java procedure, you must ensure that you have the Java code for the procedure, compile the Java code, and put the compiled code into a JAR file, and place the JAR file on the machine where the Virtual View Manager server is running.

The next section describes how to add the JAR file the server.

**Procedure**

1. Right-click at an appropriate location in the resource tree, and select **New Data Source**.
2. In the **Add Physical Data Source** window, select **Custom Java Procedure** for the driver, and click **Next**.
3. In the screen for specifying connection information, supply the relevant information, as follows:
   - **Name**—Name for the data source. When the process of adding the data source is complete, this name will be displayed in the resource tree representing the data source.
   - Use the **Browse** button to locate the path to the JAR file containing the procedure(s) on the server, or type the full path to the JAR file. This is the JAR file you prepared as described in the section “Custom Java procedures” on page 55. The JAR can be uploaded only from a file location that is visible to the server.
   - In the **Additional classpath** field, specify any classpath that might be needed by the Java custom procedure class. If no such reference to a classpath is needed, you can leave this field blank. To specify multiple CLASSPATHs, separate each CLASSPATH with a semicolon.
4. Click **Next**.
5. Expand the `<data source name>` node, and select the procedure(s) you want to add, and click **Finish**.

Custom Java Procedure JARs are exported with a full server backup, though the tool backup_export -excludejars option may be used to omit those files when required. Custom Java Procedures are normally imported into the directory conf/customjars/ when restoring from the full server backup CAR file.

---

**Adding and removing repository resources**

If at anytime after you've added a data source you want to remove a resource from the metadata repository or add more resources from the underlying data source to the repository, you can do that from IBM Cognos Virtual View Manager.

When you remove a resource in this manner, you are removing it only from the repository, not from the underlying data source itself.
Procedure

1. Right-click the desired data source in the resource tree, and select Add/Remove Resources. Alternatively, you can open the data source's Info panel, and click Add/Remove Resources (see "Obtaining information about a data source" on page 59).

   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 an IBM DB2 data source" on page 30.

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, and select or clear the Detect New Resources During Re-Introspection box. For more information, see "Adding an IBM DB2 data source" on page 30.
   - To remove a resource, clear the corresponding check box.

4. Click Finish once you have added or removed the desired resources.

Data source re-introspection

The metadata repository stores a data source exactly how you define it when adding it to IBM Cognos Virtual View Manager. The data source in the metadata repository is a representation of the underlying data source. If the underlying data source changes later on with the addition or deletion of resources, the Virtual View Manager representation of it is not modified automatically in the repository.

However, Virtual View Manager provides a mechanism called re-introspection to detect the changes in the underlying data source and notify you. You are also given the option to merge such detected changes automatically into the data source in the repository, so as to make the underlying data source and its Virtual View Manager representation to be synchronous.

The server re-introspects the underlying data source and compares its latest metadata to the metadata of its Virtual View Manager representation. Upon re-introspection, the server assembles the differences between the underlying data source and its Virtual View Manager representation, and notifies relevant recipient(s) accordingly. The server also informs if no changes are detected.

The re-introspection service assumes that a flat file data source has a header row. If the re-introspected flat file does not have a header row, the server only notifies whether there are more or less columns than the last time. If a column has been added and then deleted the server cannot detect it because there are no column headers to compare.

During re-introspection, the server searches for changes in tables and columns and updates the Virtual View Manager metadata repository, if directed so, in order to properly represent the underlying data source.

The server also logs all the changes into a list of changes and a list of affected resources. If the server did not detect any changes, it logs the occurrence of the re-introspection and notifies that there were no changes.
Changes detected during re-introspection

The re-introspection service detects any change (addition, deletion, or update) in a container only if the container and all of its contents were originally introspected and added to the metadata repository.

This is achieved by selecting the Detect New Resources During Re-Introspection option during the initial introspection (see “Adding an IBM DB2 data source” on page 30).

For details on introspecting a data source and adding it to the metadata repository, see “What it means to add data sources” on page 27.

Depending upon the type of the data source (relational or file), all changes are detected including the following:

- Changes in column data types
- Removed or added columns
- Removed or added new tables

Re-introspecting a data source

Follow these steps to re-introspect a data source.

Procedure

1. Right-click the desired data source name in the resource tree and select Open.
2. In the editor that opens on the right, click the Re-Introspection tab on the lower part of the editor.
   
   The window for re-introspection lets you introspect a data source right away or schedule the re-introspection at a specific time.
3. To re-introspect immediately, click Re-introspect Now in the Immediate Re-Introspection section.
   
   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 an IBM DB2 data source” on page 30.

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

   The Re-introspect Now command now displays a list of differences on your screen. See “Viewing re-introspection results” on page 59.
4. To schedule the re-introspection periodically, use the Scheduled Re-Introspection section.
   
   Note that the Scheduled Re-Introspection option is available only if the password was saved when the data source was originally added to the server.

   • Select Save Detected Changes if you want the changes detected in the data source during re-introspection to persist (that is, saved in the repository).

   • Select Repeat Every < > minute(s), and type the desired number of minutes in the text field.

   • Select Repeat Hourly for the re-introspection to recur every hour.

   • Select Daily for the re-introspection to recur every day.

   • 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 re-introspection will 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.

- Type and confirm the e-mail address(es) of the recipient(s) to receive the re-introspection report.
  You can specify a list of e-mail addresses in the E-mail fields by separating each one with a comma (,) or semicolon (;).

5. Save the entries.

**Viewing re-introspection results**

Follow these steps to view re-introspection results.

**Procedure**

1. When you are scheduling a re-introspection, if you want to re-introspect immediately, click Re-introspect Now in the Immediate Re-Introspection section of the re-introspection window.
   The Re-introspect Now command displays a list of differences in the Re-introspection Result window.

2. Do one of the following:
   - Click **Save** if you want the changes to be merged with the data source in the repository and be available for further use.
   - Click **Cancel** if you do not want to merge the changes with the underlying data source in the repository.

**Obtaining information about a data source**

The Info panel in the data source editor displays details about the data source connection and container definition. Different data sources will have variations on the information displayed in the Info panel. All data sources provide for review of relevant data source settings and for testing of the connection between the data source.

If you edit any of the details—such as connection information or advanced properties—in the Info panel, you must re-introspect the data source for the new properties to apply. For details on re-introspection, see “Re-introspecting a data source” on page 58.

**Procedure**

1. Right-click the desired data source name in the resource tree, and select **Open**.
2. In the editor that opens on the right, click the **Info** tab on the lower part of the editor if the **Info** panel is not already open.
   The **Info** panel has the following sections in its **Basic** panel:

*Table 4. Data source information - basic panel*

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Un-editable details about the data source which were provided when the data source was added to the Virtual View Manager server.</td>
</tr>
<tr>
<td>Name</td>
<td>User-defined name given to the data source during introspection.</td>
</tr>
</tbody>
</table>
### Table 4. Data source information - basic panel (continued)

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type of the resource, which is Data Source in this case.</td>
</tr>
<tr>
<td>Owner</td>
<td>Owner of the data source.</td>
</tr>
<tr>
<td>Owner Domain</td>
<td>Domain to which the owner of the data source belongs.</td>
</tr>
<tr>
<td>Caching</td>
<td>Shows the paths to the cache status table and cache tracking table.</td>
</tr>
<tr>
<td>Status Table</td>
<td>Tracks which views and procedures currently have cached data stored, when they were last refreshed, and so on.</td>
</tr>
<tr>
<td>Tracking Table</td>
<td>Tracks which views and procedures are currently using the data source for caching, and what tables in the data source are in use.</td>
</tr>
<tr>
<td>Connection Information</td>
<td>Connection details that are needed for the Virtual View Manager server to locate the data source and connect to it. These data source-specific details were provided when the data source was introspected, and some of them can be updated.</td>
</tr>
<tr>
<td>Annotation</td>
<td>Notes on the data source. Any user with the WRITE permission can edit the annotation. You might need to scroll down to access this field.</td>
</tr>
</tbody>
</table>

The **Info** panel includes the following details in its **Advanced** panel:

### Table 5. Data source information - advanced panel

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection pool properties</td>
<td>Options for specifying the values for the connection pool's timeout, minimum size, and maximum size properties.</td>
</tr>
<tr>
<td>Connection Validation Query</td>
<td>Enables input of a connection validation query that is sent with Test Connection use.</td>
</tr>
</tbody>
</table>
| Execute SELECTs Independently | If this option is selected, a SELECT statement submitted to this data source will be executed using a new connection from the connection pool and committed immediately after the SELECT is completed.  
  INSERT, UPDATE, and DELETE statements will continue to be executed using the same connection as part of the transaction. |
| Ignore Procedure Return Parameter | To suppress or ignore the success/failure return parameter for Sybase (and SQL) stored procedures mark the ignore procedure return parameter check box when JDBC clients do not utilize or wish to view the procedure return parameter. Otherwise a return parameter will be inserted into procedure definitions by the JDBC driver. |
| Streaming Results Mode      | If selected, this option sends the result from MySQL to the Virtual View Manager server in streams.                                         |
| Enable Data Source          | Makes the data source available for use.                                                                                                   |
| Add/Remove Resources        | Opens a window for adding/removing data source resources.                                                                                  |
| Test Connection             | Validates the connection properties. If a file-type data source is accessible through a URL, this button tests for the existence of the file at that URL. On the other hand, since a local file-type data source may contain more than one file, a check for their existence is not done when this button is clicked. |
Data source drivers

If a relational data source does not have a driver installed on the IBM Cognos Virtual View Manager metadata environment, you need to install that driver. Virtual View Manager ships with a JDBC interface and provides a bridge to a specific data source that is not currently supported. You can write a driver that connects to that interface.

You must supply the specific JDBC driver and direct the server to upload it to the system. One driver is sufficient to connect to any number of the same type of data sources. Once uploaded, the JDBC driver will function like any other JDBC driver, such as Oracle, Microsoft SQL Server, or MySQL.

The JDBC driver you supply must work properly, since the server assumes that the driver is implemented properly. The server does not make any accommodations for JDBC drivers that don’t supply correct metadata about the data source or does not retrieve result sets that are not consistent with the metadata.

For details on obtaining pre-configured JDBC drivers and installing them on the Virtual View Manager server computer, see the “Installing and Using Special JDBC Drivers” topic in the “Post-Installation Tasks” chapter of the IBM Cognos Virtual View Manager Administration Guide.

ERP data sources

IBM Cognos Virtual View Manager supports Salesforce.com, SAP, and Siebel data sources.

For more information to directly access databases from various ERP vendors using the dynamic query mode, see the IBM Cognos Dynamic Query Guide.

Adding a Salesforce.com data source

You can add Salesforce.com data sources to IBM Cognos Virtual View Manager without additional installation. If you want to use Salesforce.com functions, you must install them.

Procedure

1. In the resource tree, right-click an appropriate location, and select New Data Source.
2. In the New Physical Data Source window, select Salesforce.com, and click Next.
3. In the New Physical Data Source window, specify the following connection information.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>User-supplied name to represent the underlying Salesforce.com data source. When the process of adding the data source is complete, this name is displayed in the resource tree.</td>
</tr>
<tr>
<td>Username</td>
<td>The username required to connect to and authenticate with the Salesforce.com account to be used.</td>
</tr>
</tbody>
</table>
4. Optional: On the Advanced Connection Properties tab, additional properties are available.

Table 7. Options and descriptions of additional connection properties

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection timeout (sec)</td>
<td>Default timeout for the Salesforce.com operation.</td>
</tr>
<tr>
<td>User Concurrent Calls Limit</td>
<td>Represents the number of query cursors that can remain open at any point in time for a single user.</td>
</tr>
<tr>
<td>Concurrent Calls Limit</td>
<td>Represents the number of calls that can be executed concurrently at the organization level.</td>
</tr>
<tr>
<td>Query batch size</td>
<td>Number of records to be retrieved in a single batch.</td>
</tr>
<tr>
<td>Maximum retries</td>
<td>Maximum number of attempts to retry a failed operation.</td>
</tr>
<tr>
<td>Wait between retries (sec)</td>
<td>Number of seconds to wait between attempts to retry a failed operation.</td>
</tr>
<tr>
<td>Proxy host</td>
<td>Host or IP address of a Web proxy if one is being used.</td>
</tr>
<tr>
<td>Proxy port</td>
<td>Port number of the Web proxy if one is being used.</td>
</tr>
<tr>
<td>Proxy username</td>
<td>Username required to authenticate with Web proxy.</td>
</tr>
<tr>
<td>Proxy password</td>
<td>Password required to authenticate with Web proxy.</td>
</tr>
<tr>
<td>Web page</td>
<td>Location of the Salesforce.com environment.</td>
</tr>
</tbody>
</table>

Salesforce.com data source capabilities

IBM Cognos Virtual View Manager supports several Salesforce.com capabilities.

- **Aggregate functions** are pushed down to the Salesforce.com layer. For performance reasons, aggregate functions are pushed down to Salesforce.com by default. When queries containing aggregate functions are submitted to Salesforce.com, Cognos Virtual View Manager returns a single batch of records to satisfy the query. The maximum size of the batch is 2000 rows.
  - SUM
  - MAX
  - MIN
- AVG
- COUNT

• **Groupby** and **Orderby** clauses are pushed down to the Salesforce.com layer.
• The following date functions are supported:
  - CALENDAR_YEAR
  - CALENDAR_QUARTER
  - CALENDAR_MONTH
  - WEEK_IN_YEAR
  - WEEK_IN_MONTH
  - FISCAL_MONTH
  - DAY_ONLY
  - DAY_IN_YEAR
  - HOUR_IN_DAY

• **ConvertTimeZone**

• **Packaged Queries** - Salesforce Object Query Language (SOQL) provides a query called a "relationship query", which enables retrieving data from multiple objects in a single request. This allows Cognos Virtual View Manager to push down joins to the underlying Salesforce.com data source. Users can write relationship queries in SOQL and push them down to the Salesforce data source.

• **Single Sign-on** – Cognos Virtual View Manager supports Single Sign-on (SSO) for queries for an IBM Cognos Business Intelligence Server environment that is secured using Salesforce.com security through a custom authentication provider. For more details on achieving SSO with Salesforce.com in the Cognos Business Intelligence Server environment, please refer to IBM Developer Works.

**Adding an SAP data source**

If you are using SAP as a data source for IBM Cognos Virtual View Manager, you must ensure that you have the SAP Java Connector Library (SAP JCo) installed and configured. Licensed users of SAP can download the SAP JCo library from SAP Service Marketplace.

You must have a 32-bit compliant version of the SAP JCo library installed on the same machine that is running the Cognos Virtual View Manager Server.

**Before you begin**

Install the SAP Java Connector Library for your operating system. For more information, see "Install the SAP Java Connector Library" on page 65.

**Procedure**

1. In the resource tree, right-click an appropriate location, and select **New Data Source**.
2. In the **New Physical Data Source** window, select **SAP** and click **Next**.
3. In the **New Physical Data Source** window, specify the properties described in the following table.
### Table 8. Properties and descriptions from the New Physical Data Source window

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>User-supplied name to represent the SAP data source. After the data source is added, this name is displayed in the resource tree.</td>
</tr>
<tr>
<td><strong>Application Server</strong></td>
<td>The host of the SAP environment you are accessing. For load balanced SAP environments, this entry is not required.</td>
</tr>
<tr>
<td><strong>SAP Router String</strong></td>
<td>This value is required if you are accessing a router to connect to the SAP application server.</td>
</tr>
<tr>
<td><strong>System Number</strong></td>
<td>System number of the SAP system that you are connecting to.</td>
</tr>
<tr>
<td><strong>Client</strong></td>
<td>Client number of the SAP system that you are connecting to.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>The username required to connect to and authenticate with the SAP system.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The password required to connect to and authenticate with the SAP system.</td>
</tr>
<tr>
<td><strong>Pass-through login</strong></td>
<td>Necessary to enable single sign-on (SSO) in Cognos Business Intelligence environments that are secured using SAP security. Cognos Virtual View Manager supports (SSO) for queries for a Cognos BI Server environment that is secured using SAP security via a SAP BW namespace.</td>
</tr>
</tbody>
</table>

4. Optional: On the **Advanced Connection Properties** tab, the additional properties described in the following table are available.

### Table 9. Properties and descriptions from the Advanced Connection Properties tab

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum connections in pool</strong></td>
<td>This allows users to configure limits for the connection pool that is created.</td>
</tr>
<tr>
<td><strong>Table row count</strong></td>
<td>This is used to control the number of rows returned by SAP for table queries.</td>
</tr>
<tr>
<td><strong>ABAP Query DB access limit</strong></td>
<td>This is used to limit the number of accesses during query execution.</td>
</tr>
<tr>
<td><strong>Table read function</strong></td>
<td>Users can choose to implement their own table read function. RFC_TABLE_READ is set by default. If users specify a custom table read function like YRFC_TABLE_READ and if the data field length for that function is different than 512, then the same needs to be updated in the <code>VVM\apps\dlm\cis_ds_sap\conf\sapadapterconfig.xml</code> file. Also, this file has the option to parse decimal fix by setting the parseDecimalValues property value to true.</td>
</tr>
<tr>
<td><strong>Message server</strong></td>
<td>This property is required for connecting to SAP in a load balanced environment.</td>
</tr>
<tr>
<td><strong>System ID</strong></td>
<td>This property is required for connecting to SAP in a load balanced environment.</td>
</tr>
<tr>
<td><strong>Logon Group</strong></td>
<td>This property is required for connecting to SAP in a load balanced environment.</td>
</tr>
</tbody>
</table>
Table 9. Properties and descriptions from the Advanced Connection Properties tab (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date Format</strong></td>
<td>This property can be used to configure a user specific date format. For example: DD:MM:YYYY or YYYY-MM-DD.</td>
</tr>
<tr>
<td><strong>Time Format</strong></td>
<td>This property can be used to configure user specific time format. For example: HH:MM:SS.</td>
</tr>
</tbody>
</table>

Install the SAP Java Connector Library

You must install the SAP Java Connector Library (SAP JCo), versions 2.1.8 or later, on the same computer as IBM Cognos Virtual View Manager software. There are separate installation procedures for Windows, Linux, Solaris, AIX®, and HP-UX operating systems.

SAP JCo is available at the SAP Service Marketplace at www.sap.com. To download, you need to be a registered customer. When you log in, navigate to SAP Java Connector > Tools and Services, and download the JCo distribution for the platform on which you are running Cognos Virtual View Manager.

To avoid a conflict between JCo versions, you should not install Cognos Virtual View Manager Application Data Services for SAP on a computer already installed with the SAP GUI.

Important:

Different JCo libraries exist for specific processor manufacturers and architectures. The JCo library you install must work with the processor manufacturer and architecture in the environment in which Virtual View Manager is installed. For example, if you use an Intel processor and a 32-bit system for Cognos Virtual View Manager, be sure to install the JCo library that works with this configuration.

Installing SAP JCo on Windows computers:

You must install the SAP Java Connector Library (SAP JCo) for a Microsoft Windows operating system on the same computer as IBM Cognos Virtual View Manager.

Before you begin

Verify that the Windows System 32 directory includes the files MSVCR71.DLL and MSVCP71.DLL. These DLL files are Shared C Runtime (CRT) components that are required by SAP JCo but that are not shipped with SAP JCo. SAP JCo and Cognos Virtual View Manager Application Data Services for SAP requires these DLL files. If these DLL files are missing, install the Microsoft .NET Framework SDK Version 1.1. For more information, see the Microsoft Developer Network (MSDN) download site.

Procedure

1. Extract the sapjco-ntintel-2.x.xx.zip file to a temporary directory.
2. Copy the sapjco.jar file to the <VVM>jre\lib\ext directory, where <VVM> is the root directory of your Cognos Virtual View Manager server.
3. Copy the sapjcorfc.dll and 1ibrfc32.dll files to the Windows System32 directory. Typically, this directory is C:\Windows\System32.
4. For information on testing the SAP JCo installation for a Microsoft Windows operating system, see the SAP Java Connector documentation.

**Install SAP JCo on UNIX or Linux Computers:**

You must install the SAP Java Connector Library (SAP JCo) for UNIX or Linux operating systems on the same computer as IBM Cognos Virtual View Manager.

For information on testing and verifying the SAP JCo installation for UNIX or Linux operating systems, see the SAP Java Connector documentation.

**Installing SAP JCo on Linux:**

The SAP Java Connector Library can be installed on a computer that is running a Linux operating system.

**Procedure**

1. Extract the sapjco-linuxintel-2.x.xx.tgz file to a temporary directory.
2. Copy the sapjco.jar file to the `<VVM>/jre/lib/ext` directory, where `<VVM>` is the root directory of your Cognos Virtual View Manager server.
3. Copy the `librfccm.so` and `libsapjcorfc.so` files into the `<VVM>/jre/lib/i386` directory, where `<VVM>` is the root directory of your Cognos Virtual Manager server.

**Installing SAP JCo on Solaris:**

The SAP Java Connector Library can be installed on a computer that is running a Solaris operating system.

**Procedure**

1. Extract the sapjco-sun-2.x.xx.tgz file to a temporary directory.
2. Copy the sapjco.jar file to the `<VVM>/jre/lib/ext` directory, where `<VVM>` is the root directory of your Cognos Virtual View Manager server.
3. Copy the `librfccm.so` and `libsapjcorfc.so` files into the `<VVM>/jre/lib/sparc` directory, where `<VVM>` is the root directory of your Cognos Virtual Manager server.

**Installing SAP JCo on AIX:**

The SAP Java Connector Library can be installed on a computer that is running an AIX operating system.

**Procedure**

1. Extract the sapjco-rs6000-2.x.xx.tgz file to a temporary directory.
2. Copy the sapjco.jar file to the `<VVM>/jre/lib/ext` directory, where `<VVM>` is the root directory of your Cognos Virtual View Manager server.
3. Copy the `librfccm.so` and `libsapjcorfc.so` files into the `<VVM>/jre/lib/ppc` directory, where `<VVM>` is the root directory of your Cognos Virtual Manager server.

**Important:** In some SAP JCo versions, the `libsapjcorfc.so` file is incorrectly named. Make a copy of the incorrectly named file `libsapjcorfc.o` and rename the copy of the file to `libsapjcorfc.so`. 
**Installing SAP JCo on HP-UX:**

The SAP Java Connector Library can be installed on a computer installed that is running an HP-UX operating system.

**Procedure**

1. Extract the `sapjco-hp_32-2.x.xx.tgz` file to a temporary directory.
2. Copy the `sapjco.jar` file to the `<VVM>/jre/lib/ext` directory, where `<VVM>` is the root directory of your Cognos Virtual View Manager server.
3. Copy the `librfccm.sl` and `libsapjcorfc.sl` files into the `<VVM>/jre/lib/PA_RISC2.0` directory, where `<VVM>` is the root directory of your Cognos Virtual Manager server.

**SAP data source resources**

IBM Cognos Virtual View Manager enables users to introspect metadata from and query against supported SAP systems.

The following SAP resources can be introspected and queried using Cognos Virtual View Manager:

- ABAP Queries
- Infosets
- RFCs
- BAPIs
- SAP Tables

**SAP tables**

SAP tables provide specific SQL support and capabilities.

**SQL support for SAP tables:**

Each column in a SAP table is mapped to a column of an introspected IBM Cognos Virtual View Manager table.

Primary key columns are also indicated as primary keys in the introspected table. Cognos Virtual View Manager supports the data types that are supported by the SAP table read function (RFC_READ_TABLE).

The following table displays the data type supported by SAP tables.

**Table 10. SQL support for SAP tables**

<table>
<thead>
<tr>
<th>ABAP Type</th>
<th>VVM Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C,N,X,g</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER</td>
</tr>
<tr>
<td>P</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>F</td>
<td>FLOAT</td>
</tr>
<tr>
<td>D</td>
<td>DATE</td>
</tr>
<tr>
<td>T</td>
<td>TIME</td>
</tr>
</tbody>
</table>
Limitations:

Columns whose length exceeds 512 bytes are automatically ignored, at the time of introspection, as these columns cannot be queried using the default table read function "RFC_READ_TABLE".

Capabilities and SAP tables:

The following table lists the SAP capabilities supported by IBM Cognos Virtual View Manager.

**Table 11. SAP capabilities supported by Cognos Virtual View Manager**

<table>
<thead>
<tr>
<th>Capability</th>
<th>Supported in Queries</th>
<th>Pushed to SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter (where clause)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IN clause</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BETWEEN</td>
<td>Yes</td>
<td>Between clause is converted to combination of &lt;= and &gt;= operators and pushed to SAP</td>
</tr>
<tr>
<td>LIKE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NULL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NOT NULL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Function cast</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

All standard SQL operators such as >, <, >=, <=, <>, =, are supported by Cognos Virtual View Manager in queries against SAP Tables.

By default, Cognos Virtual View Manager returns a maximum batch of 30000 rows for a table query. This value is controlled by a property called sapTableQueryBatchSize which is stored in the 'sapadapterconfig.properties' file, located at: \\apps\dlm\cis_ds_sap\conf.

Cognos Virtual View Manager uses sapTableQueryBatchSize to process all requests in batches. This property can be used to fine tune adapter performance. If there is no value, Cognos Virtual View Manager uses the default configured value of 30000.

**SAP functions**

SAP functions include parameter mapping, capabilities, and SAP functions.

**SAP parameter mapping:**

To invoke a SAP function, you must provide the function name, function import parameters, export parameters, and table parameters.

IBM Cognos Virtual View Manager maps all import, export, and table parameters to columns in a virtual table to represent a SAP function. The following naming convention is used during the mapping process:

- Import parameter: <Import><SPACE><parameter name>
- Export parameter: <Export><SPACE><parameter name>
- Table parameter: <Table Name><SPACE><parameter name>

In the case of structure parameters:
SAP functions and capabilities:

SAP functions support the “EQUALS” operator in a filter. If any other operator is used, an Operator not supported error exception is generated.

This table summarizes SAP functions and capabilities.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Supported in Queries</th>
<th>Pushed to SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter (where clause)</td>
<td>Yes*</td>
<td>Yes</td>
</tr>
<tr>
<td>IN clause</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>OR clause</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>Function cast</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

* see Parameter Mapping

SAP functions do not support standard SQL operations. The IN clause, and the logical operators ‘OR’ and ‘AND’ are invoked by remote method calls and take on a special meaning.

- **OR** is used as a new function call.
  - For example, A=10 OR A=20. In this case IBM Cognos Virtual View Manager calls the function twice; first with A=10 and the second time with A=20. Results from both calls are joined later.
- **AND** is used to combine filters on different columns.
  - For example, A=10 and B=20. In this case, Cognos Virtual View Manager only calls the function once with filters A=10 and B=20.

**IN clause**

- **IN** clause with import parameters: If an IN clause is used with an import parameter, it is treated as a series of OR expressions.
  - For example:
    - Select * from BAPI_EMP_GELIST where Import EMP_ID IN (‘1’,’2’,’3’) 
      - This is equivalent to:
        - Select * from BAPI_EMP_GELIST where (Import EMP_ID=’1’ OR Import EMP_ID=’2’ OR Import EMP_ID=’3’) 

- **IN** clause with table parameter: If an IN clause is used with a table parameter, it is treated as a set of input rows for a given column.
  - For example: Querying a BAPI where DEPT_TABLE is used for import
    - Select * from BAPI_EMP_GELIST where DEPT_TABLE DEPT_ID IN (‘1’,’2’,’3’) 
    - First row -> DEPT_ID=’1’
    - Second row -> DEPT_ID=’2’
    - Third row -> DEPT_ID=’3’
  - For multiple invocations:
    - Select * from BAPI_EMP_GELIST where DEPT_TABLE DEPT_ID IN (‘1’,’2’) and DEPT_TABLE DEPT_MGR_ID IN (‘8’,’9’) 
    - First row DEPT_ID=’1’ and DEPT_TABLE MGR_ID=’8’
Second row DEPT_ID='2' and DEPT_TABLE MGR_ID='9'

**Result mapping**
Result mappings in IBM Cognos Virtual View Manager include import, export, and table parameters.

**Import and export parameters:**
These scalar values are returned unmodified in the result and are repeated if there is more than one row.

Import parameters are used to assign values to the import parameter of a user-defined function, or to the import parameters of standard function at runtime.

**Table parameters:**
BAPIs and RFCs can contain more than one table.

IBM Cognos Virtual View Manager allows users to select fields from one table at a time in the query against SAP BAPI or RFC. If fields from more than one table structure are included in the query, Cognos Virtual View Manager generates an exception.

**SAP ABAP queries and infosets**

**Data type mapping:**
The following table displays the data type supported by ABAP Queries.

<table>
<thead>
<tr>
<th>ABAP Type</th>
<th>VVM Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, N</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER</td>
</tr>
<tr>
<td>P</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>F</td>
<td>FLOAT</td>
</tr>
<tr>
<td>D</td>
<td>DATE</td>
</tr>
<tr>
<td>T</td>
<td>TIME</td>
</tr>
</tbody>
</table>

**SQL capabilities:**
The following table lists the SQL capabilities supported by IBM Cognos Virtual View Manager.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Supported in Queries</th>
<th>Pushed to SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter (where clause)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IN clause</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BETWEEN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NULL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IS NOT NULL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Function cast</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Supported operators:

All standard SQL operators such as >, <, >=, <=, <>, = are supported in queries against ABAP queries.

Adding a Siebel data source

You can add Siebel data sources to IBM Cognos Virtual View Manager without additional installation.

Before you begin

Ensure you have installed the Siebel Data Bean JARs. For more information, see “Installing the Siebel Data Bean JAR files” on page 72.

Procedure

1. In the resource tree, right-click at appropriate location and select New Data Source.
2. In the New Physical Data Source window, select Siebel and click Next.
3. In the New Physical Data Source window, specify the following connection information.

Table 15. Properties and descriptions from the New Physical Data Source window

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>User-supplied name to represent the underlying Siebel data source. When the process of adding the data source is complete, this name will be displayed in the resource tree.</td>
</tr>
<tr>
<td>Gateway Name Server</td>
<td>Coordinates the Siebel Enterprise Server and Siebel servers in the environment.</td>
</tr>
<tr>
<td>Gateway Port</td>
<td>Port that Gateway Name Server listens on.</td>
</tr>
<tr>
<td>Siebel Enterprise Name</td>
<td>Name of the Siebel Enterprise Server</td>
</tr>
<tr>
<td>Application Object Manager</td>
<td>Name of the Siebel Application Object Manager.</td>
</tr>
<tr>
<td>Language</td>
<td>Siebel instance language.</td>
</tr>
<tr>
<td>Username</td>
<td>The username required to connect to and authenticate with Siebel.</td>
</tr>
<tr>
<td>Password</td>
<td>The password required to connect to and authenticate with Siebel.</td>
</tr>
<tr>
<td>Pass-through login</td>
<td>Cognos Virtual View Manager supports Single Sign-on (SSO) for queries for a Cognos BI Server environment that is secured using Siebel security via a custom authentication provider. For more details on achieving SSO with Siebel in the Cognos BI Server environment, please visit IBM Developer Works.</td>
</tr>
</tbody>
</table>

4. Optional: On the Advanced Connection Properties tab, additional properties are available.

Table 16. Properties and descriptions from the Advanced Connection Properties window

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Repository</td>
<td>Specifies the name of the Siebel Repository from which to fetch the metadata. This property is required for introspection.</td>
</tr>
</tbody>
</table>
Table 16. Properties and descriptions from the Advanced Connection Properties window (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>Specifies the version of Siebel software installed on the server.</td>
</tr>
<tr>
<td>CodePage</td>
<td>This setting specifies the client codepage to the Siebel server.</td>
</tr>
<tr>
<td>Transport</td>
<td>Specifies the transport protocol to be used by the Siebel Data Bean API for communication with the Siebel server.</td>
</tr>
<tr>
<td>Compression</td>
<td>This setting specifies the compression algorithm to be used by the Siebel Data Bean API for communication with the Siebel server.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Specifies the encryption algorithm to be used by the Siebel Data Bean API when communicating with the Siebel server.</td>
</tr>
<tr>
<td>Login Retries</td>
<td>Specifies the number of times to retry upon initial login failure.</td>
</tr>
<tr>
<td>Login Retry Delay</td>
<td>Specifies the wait period between login retries.</td>
</tr>
<tr>
<td>Maximum Records</td>
<td>The maximum number of rows of data to be fetched for any given data query.</td>
</tr>
<tr>
<td>View Mode</td>
<td>Specifies the View Mode to fetch data. This provides the ability to achieve role based access to data.</td>
</tr>
</tbody>
</table>

Installing the Siebel Data Bean JAR files

You must install the Siebel Data Bean JAR files to enable connectivity to Siebel.

Before you begin

If you do not have access to the Siebel Data Bean Jar files see your Siebel System Administrator.

About this task

The following instructions describe how to install the Siebel Data Bean JAR files for Siebel versions 8.0 and 8.1.

Procedure

1. Create a directory for the Siebel Data Bean JAR files at the following location `<VVM>/apps/dlm/app_ds_siebel/lib/<version>`, where `<VVM>` is the root directory of your Virtual View Manager Server and `<version>` is the Siebel version (8.0 or 8.1).
2. Copy the `Siebel.jar` and `SiebelJI_enu.jar` files obtained from your Siebel Administrator to the directory that you created in Step 1.

Siebel data sources

Siebel data sources map to specific objects in IBM Cognos Virtual View Manager and support specific SQL capabilities.

Siebel object mapping in Cognos Virtual View Manager:

IBM Cognos Virtual View Manager enables users to introspect Siebel Business Objects, Business Components, Business Services, and Business Methods.
Introspected Siebel objects are mapped to Cognos Virtual View Manager objects as follows:

<table>
<thead>
<tr>
<th>Siebel Object</th>
<th>Cognos Virtual View Manager Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siebel Business Objects</td>
<td>Cognos Virtual View Manager Schema</td>
</tr>
<tr>
<td>Siebel Business Components</td>
<td>Cognos Virtual View Manager Tables</td>
</tr>
<tr>
<td>Siebel Business Services</td>
<td>Cognos Virtual View Manager Schema</td>
</tr>
<tr>
<td>Siebel Business Methods</td>
<td>Cognos Virtual View Manager Procedures</td>
</tr>
</tbody>
</table>

**Siebel SQL capabilities:**

The following table lists the SQL capabilities supported by IBM Cognos Virtual View Manager.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Supported in Queries</th>
<th>Pushed down to Siebel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filters</td>
<td>Yes</td>
<td>WHERE clause passed to Siebel as a Search Expression.</td>
</tr>
<tr>
<td>Filter (IN)</td>
<td>Yes</td>
<td>Passed to Siebel as a series of OR clauses.</td>
</tr>
<tr>
<td>Filter (BETWEEN)</td>
<td>Yes</td>
<td>Passed down to Siebel as an expression of &gt;= and &lt;. For example: <em>x BETWEEN a,b</em> is passed as <em>x &gt;=a AND x &gt; b</em>.</td>
</tr>
<tr>
<td>Filter (LIKE)</td>
<td>Yes</td>
<td>Passed down to Siebel as LIKE, case insensitive by default.</td>
</tr>
<tr>
<td>Join</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Group By</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Aggregate Functions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cast Functions</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Order By</td>
<td>Yes</td>
<td>Passed to Siebel as sort Expressions. The default sort is descending.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CASE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Subquery</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>UNION</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Cognos Virtual View Manager does not support the DML functions INSERT, UPDATE, and DELETE.
Chapter 4. Views

This chapter describes how to build and execute a view. You should be familiar with the information presented in the preceding chapters.

About views

A view is a virtual data table defined by SQL and IBM Cognos Virtual View Manager metadata.

Because the view is defined as a resource, it can SELECT, JOIN, UNION, and more from any available data sources defined in the Virtual View Manager data layer. Views can be selectively published to make them available to end users using client programs. A view is designated in the resource tree with the icon.

The following sections describe the components of the view editor and how to create and work with views.

Note: If you want to make a view available to client programs through a JDBC, ODBC, ADO.NET, or SOAP connection, you must publish that view.

- For details on publishing resources, see the chapter Publishing Resources in this guide.
- For details on caching a view and planning a view's execution, see Performance Tuning.
- For information on scheduling a view's execution, see the section Creating a Timer Event Trigger.

The view editor

IBM Cognos Virtual View Manager provides a view editor for building and editing views. This editor opens automatically when you create a view.

At any time later, you can open a view's editor by double-clicking the view's name in the resource tree. The following paragraphs describe the panels and the toolbar available in the view editor.

View editor panels

The view editor has specific panels, displayed in the right pane, for viewing and editing the various properties of a view. You can open each of these panels by clicking the corresponding tab located at the bottom of the pane.

Each of these is introduced and briefly described in the following sections.

Model panel

Use this panel for adding resources to a view when designing the query.

In the Model panel, you can do the following:
- Include tables (any type, such as relational, LDAP, delimited file) and procedures that contain either all scalar outputs or only one cursor output. You can drag multiple resources from the resource tree and drop them into the Model panel.
To select adjacent resources, keep the Shift key pressed and select the resources. To select non-adjacent resources, keep the Ctrl key pressed and select the resources.

- Join columns between tables, and specify the join algorithms and cardinality.
- Eliminate duplicate rows by specifying the DISTINCT global query option.
- Specify global query hints.

For details on using the **Model** panel, see "Tasks involved in designing views" on page 81.

The **Model** panel works in conjunction with the Grid panel, which is described in the section "Grid panel."

### Grid panel

Use the this panel for projecting columns that are either included in the SQL execution results or constrain the query (or both). The output columns are the ones that are ultimately published as Virtual View Manager database tables or Web service operations for JDBC/ODBC client accessibility.

In the **Grid** panel, you can do the following:

- Specify columns for projection.
- Move the columns up or down in order to determine their output order.
- Supply aliases for column names.
- Sort columns in ascending or descending order.
- Specify the GROUP BY option for a query.
- Specify query constraints that constitute the WHERE clause of the SQL.
- Add functions and declare variables for the SQL.

This panel works in conjunction with the **Model** panel in the sense that if no resources are included in the **Model** panel, no tables will be listed in the **Grid** panel to select columns, and the query will be invalid.

For details on using the **Grid** panel, see "Tasks involved in designing views" on page 81.

The following sections describe the use of each column in the **Grid** panel.

#### Move up/down

In this column, you can select a row and move it up/down either by using the **Move Up** or **Move Down** arrow or by dragging and dropping the row up or down. By doing so, the corresponding row in the **Column** column is moved.

#### Table and column

The **Table** column in the **Grid** panel lets you specify the tables from which you can select the columns for inclusion in your view. This feature is helpful when you have several tables with numerous columns and you want to save time on selecting specific columns from specific tables.

The **Column** column in the **Grid** panel lets you select the columns that you want to include in your view.
**Alias**

The **Alias** column in the **Grid** panel is used for providing an alias for the column you select for inclusion in the view.

Alias is useful especially for providing meaningful names to result columns. The alias you supply is the name users will see when they browse or query the Virtual View Manager services you create.

**Output**

When building a view you may wish to project only specific columns in the view execution result. The **Output** column in the **Grid** panel lets you specify the columns that you want to include in your view execution result.

**Sort type and sort order**

These two columns in the **Grid** panel function together to generate the ORDER BY clause in the SQL of the current view.

**Group by**

The **Group By** column in the **Grid** panel is used for generating the GROUP BY clause in the SQL of the current view.

**Criteria and or**

The **Or** columns in the **Grid** panel are used in conjunction with the **Criteria** column. An entry in the **Criteria** column and the corresponding entry in the next **Or** column form the WHERE clause in the current view’s SQL.

**SQL panel**

Use this panel for viewing or modifying a view’s SQL.

Clicking the **SQL** tab in the view editor displays the **SQL** panel which shows the SQL statement corresponding to the view being designed in the **Model** panel.

The SQL statement is automatically generated when you construct a view using the **Model** and **Grid** panels.

In the **SQL** panel, you can do the following:

- Edit the auto-generated SQL
- Import an existing SQL and modify it as needed
- Drag a resource from the resource tree and drop it into this panel, and the path string to the resource will be placed at the cursor level.

You can also enter valid SQL statements in this panel to refine your view. But if you manually edit the SQL in the **SQL** panel, the **Model** and **Grid** panels will be disabled. Later on you can regenerate the design in the **Model** panel by using the **Generate Model** toolbar button. The columns you include in the SELECT statement in this panel must exactly match the columns designed in the **Columns** panel, including the order in which they are provided.

For details on using the **SQL** panel, see “Designing a view’s SQL in the **SQL** panel” on page 86.
**Columns panel**

This panel displays the columns that are selected in the Grid panel. The JDBC data type and the corresponding native data type of each column is also displayed.

In the Columns panel, you can do the following:
- Learn about the output columns included in a view.
- View the data type of a given column
- Design the interface of a view by adding new columns

The Columns panel works in conjunction with the SQL panel. For details, see Design mode check box.

**Design mode check box**

The Design Mode check box lets you design the projections. Designing projections using the design mode lets you do a top-to-bottom design in 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 that the columns defined in the SELECT statement of the SQL and those that are designed in the Columns panel using the design mode exactly match including the order in which they are provided; otherwise, the query will generate an error when executed.

**Add button**

The Add button lets you add new columns when you design the SQL for a view. This button is enabled when the Design Mode option is selected in the Columns panel.

**Indexes panel**

This panel lets you create indexes on tables.

When you publish a view or data source that has an index, the index is made available to JDBC/ODBC clients at /services/databases/<data source>/<view>.

For details on using this panel, see Identifying Existing Indexes.

**Foreign keys panel**

This panel lets you create specify foreign keys on tables.

For details on using this panel, see Defining Foreign Keys.

**Caching panel**

This panel lets you create a cache for the view.

For details on using this panel, see Chapter 11, “Caching resources,” on page 193.

**Info panel**

This panel displays the details about the current view.

The details are:
• Name — user-provided name and location.
• Type — resource type.
• Owner — owner name.
• Owner Domain — domain to which the owner of the view belongs.
• Annotation — annotation on the resource.

Only the annotation can be edited by a user with the WRITE privilege on the view.

Rebind panel
This panel displays a new panel at the bottom of the pane. It lets you view the
schema of the resources that a view depends upon, and also rebind a view to a
different resource if necessary.

This panel displays when you click the Show Rebind Panel button on the editor toolbar.

For details on using this panel, see “Rebinding a view” on page 89.

Result panel
By default, this panel displays the results of a view execution in 50 rows at a time.

The Result panel is displayed at the bottom of the pane when you click the
Execute button on the editor toolbar.

The Cancel button displays in the Result panel during query execution. You can use the Cancel button to cancel an in-progress execution. Also, if you close the Result panel, any in-progress view execution will be cancelled.

For details on executing a view, see “Executing a view” on page 88.

View editor toolbar

This section describes the buttons in the view editor toolbar.

The following table lists the view editor's toolbar buttons (in alphabetical order) and their functionality:

<table>
<thead>
<tr>
<th>Button</th>
<th>Label</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add</td>
<td>Opens a window to add a resource to a view.</td>
</tr>
<tr>
<td></td>
<td>Toggle Syntax Highlighting</td>
<td>Changes the color of SQL keywords used in the SQL statement.</td>
</tr>
<tr>
<td></td>
<td>Center Graph</td>
<td>Adjusts a view's dependency graph display to center around a selected resource.</td>
</tr>
<tr>
<td></td>
<td>Collapse Dependencies</td>
<td>Collapses a selected resource's dependencies in the graph display.</td>
</tr>
<tr>
<td></td>
<td>Collapse Dependents</td>
<td>Collapses a selected resource's dependents in the graph display.</td>
</tr>
<tr>
<td>Button</td>
<td>Label</td>
<td>Functionality</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| ![Button](image) | Cycle I/O Direction          | Cycles the input/output directions.  
• Blue arrow indicates an input parameter.  
• Red arrow indicates an output parameter. |
| ![Button](image) | Delete                       | Deletes a selected resource from the model.                                                                                                 |
| ![Button](image) | Execute                      | Executes the SQL of a view.                                                                                                                   |
| ![Button](image) | Find                         | Opens a search field in the lower section of the editor where you can type the desired text to be searched in the editor.  
This button also closes an opened search field. |
| ![Button](image) | Format Query                 | Formats the SQL.                                                                                                                             |
| ![Button](image) | Generate Dependency          | Updates a view's dependency graph/tree display.                                                                                               |
| ![Button](image) | Generate Model               | Generates the design in the Model panel for the SQL that has been generated from a model.                                                  |
| ![Button](image) | Insert from File             | Opens a window for specifying a location from which to upload the SQL or SQL script for a view.                                             |
| ![Button](image) | Move Parameter Down          | Moves a column down.                                                                                                                         |
| ![Button](image) | Move Parameter In            | Moves a column inward.                                                                                                                        |
| ![Button](image) | Move Parameter Out           | Moves a column outward.                                                                                                                        |
| ![Button](image) | Move Parameter Up            | Moves a column up.                                                                                                                             |
| ![Button](image) | Navigate                     | Opens the Navigator window.                                                                                                                   |
| ![Button](image) | Open Resource                | Opens a selected resource in a view's dependency graph/tree display.                                                                                |
| ![Button](image) | Properties                   | Opens the Properties window for a selected resource in a view's dependency graph/tree display.                                             |
| ![Button](image) | Rebind                       | Rebinds a view to a different resource.                                                                                                          |
| ![Button](image) | Save to File                 | Opens a window for specifying a location to save the SQL of a view.                                                                                     |
Creating new views

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 Where Can You Create a Resource?

Procedure

1. Right-click at an appropriate location in the resource tree, and select New View.
   Alternatively, at the desired location you can select File > New > View.
2. In the Input dialog, type a name for the view, and 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.
   Now you’re ready to design the view. See Tasks involved in designing views

Tasks involved in designing views

Designing a view involves performing several tasks, some of which are optional according to your needs.

The tasks are:
- Adding resources for the view
- Joining columns between tables
- Specifying join properties
- Specifying global query options
- Specifying query hints
- Selecting columns to be included in the view execution result

<table>
<thead>
<tr>
<th>Button</th>
<th>Label</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Show Contents" /></td>
<td>Show Contents</td>
<td>When a table is selected in the Model panel, it displays all the columns with data. When you right-click a table/view in the tree and select this button, it displays all the columns with data.</td>
</tr>
<tr>
<td><img src="image" alt="Show Dependency Panel" /></td>
<td>Show Dependency Panel</td>
<td>Displays the dependencies of a view and the hierarchical relationship among the dependencies.</td>
</tr>
<tr>
<td><img src="image" alt="Show Execution Plan" /></td>
<td>Show Execution Plan</td>
<td>Displays the execution plan — how Virtual View Manager server executes the SQL of a particular view. For details, see SQL Execution Plan</td>
</tr>
<tr>
<td><img src="image" alt="Show Rebind Panel" /></td>
<td>Show Rebind Panel</td>
<td>Opens the Rebind panel.</td>
</tr>
<tr>
<td><img src="image" alt="Switch Graph/Tree View" /></td>
<td>Switch Graph/Tree View</td>
<td>Toggles between the graphical and tree displays of a view’s dependencies.</td>
</tr>
</tbody>
</table>
Supplying aliases for column names
Specifying the sorting order for columns
Specifying the GROUP BY option
Specifying query constraints for the WHERE clause

These tasks can be accomplished by using either the Model panel and Grid panel or the SQL panel and Columns panel. The following sections describe how to perform these tasks.

---

**About designing a view in the Model panel**

The Model panel is a graphical tool that is often used to select the data resources for the view and join the tables. You can also change the display to more easily navigate among the tables, specify a global query option, and specify query hints. Each of these tasks is described below.

**Adding resources to the Model panel**

Use the following steps to add resources to the Model panel. These resources can be database tables, text files, procedures, and other views.

Note that only a procedure that contains either all scalar outputs or just one cursor output can be included in a view.

**Procedure**

1. Locate the desired resources in the resource tree.
2. Drag and drop the resources one by one onto the Model panel. Alternatively, you can click the Add button in the toolbar to open a window and select the desired resources.

**Joining tables in the Model panel**

Use the following steps to join tables in the Model panel.

**Procedure**

1. Select the desired column from one table, keep the left-mouse-button pressed, and join the column with an appropriate column in a different table. For details on joins and specifying join properties, see Performance Tuning.
2. Make sure that the columns are of compatible data types. To see the data type of a column, click the Columns tab and open the Columns panel. In the Model panel, the join between the selected tables is visible as a line connecting those tables (see the next screen). This type of join is called an INNER JOIN.
   For details on joins and specifying join properties, see Performance Tuning.

**Navigating among the 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 time. You can adjust the visible area using the scroll bars.

A more direct way of moving the visible area of the Model panel is available by means of the Navigator window.
Procedure

Open the navigator window by clicking the navigator button located at the lower right corner of the Model panel. The Navigator window opens. A blue rectangle in the Navigator defines the visible area in the Model panel. You can move this rectangle around to view specific areas in the Model panel.

Specifying a global query option in the Model panel

You can set the SELECT DISTINCT global option for the SQL via the Model panel.

Procedure

Right-click anywhere in the Model panel and choose Select Distinct. Such selection would add DISTINCT to the view’s SELECT statement as in the following example:

SELECT DISTINCT * FROM /shared/examples/ds_inventory/productsproducts

Specifying query hints in the Model panel

Query hints are instructions that you can include in your SQL statement for the optimizer. You can specify several global query hints for the current view.

You can specify the following query hints in the Model panel:

- Maximum number of rows to be fetched
- Case sensitivity
- Ignore Trailing Spaces
- Force disk
- Strict

Procedure

1. Add the desired number of tables in the Model panel.
2. Right-click anywhere in the Model panel, and select Query Hints. The Query Hints window displays.
3. 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 the comparisons in case-sensitive mode.
   - False — This option sets the comparisons in non-case-sensitive mode.
5. Select one of the following options in the Ignore Trailing Spaces drop-down list:
   - Server Default — Default setting of the server.
   - True — This option sets the comparisons to ignore trailing spaces.
   - False — This option does not ignore trailing spaces.
6. Select the desired check box:
   - Force Disk — If selected, this option forces the query engine to use disk instead of memory wherever possible.
   - Strict — If this option is selected, mathematical functions are not pushed to adhere to strict SQL 92 behavior. This may affect performance.
7. You can use the Clear All button to clear all the entries.
8. Click OK after setting the options.
9. Click the SQL tab to view the 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/productsproducts
```

### About designing views in the Grid panel

This panel is often used to refine the definitions of columns, sorting, and other criteria for the model.

The following sections describe how to perform these tasks using the Grid panel:
- Add a column
- Define a column alias
- Include a column in an ORDER BY clause
- Specify the GROUP BY clause
- Include criteria on a column
- Include a function
- Declare a variable

#### Adding columns in the Grid panel

Use the following steps to add a column in the Grid panel.

**Procedure**
1. Make sure that the desired tables are included in the Model panel.
   See [Adding resources to the Model panel](#).
2. Click the Grid tab to open the Grid panel.
3. Click a cell in the first column (the Column section) in the grid and select a column from the drop-down list.
   The drop-down list contains the complete collection of columns for all the tables in the model.
   Notice that the check box in the Output column is selected, which means that the column will be included in the view execution result and will be displayed in the Result panel. Deselect a box in the Output column if you want to exclude it from the view execution result.

#### Giving aliases to columns in the Grid panel

Use the following steps to give an alias to a column in the Grid panel.

**Procedure**
1. Add the desired column to the grid, as described in [Adding columns in the Grid panel](#).
2. Enter an alias for this column in the corresponding Alias cell.
   The alias you supply here is the name users will see when they browse or query the IBM Cognos Virtual View Manager data services you create, so use an alias that will make the most sense to the business application and users being served.
   If you happen to specify a reserved keyword (like column, count, or group) or a function names for an alias then it will be enclosed in double quotes in the
SQL statement. For a list of reserved words, see the end of the SQL Support chapter in the IBM Cognos Virtual View Manager Reference Guide.

Automatically generated SQL from some ODBC clients (Excel and others) will need to be modified should the alias be used to query the data sources.

### Including columns in the ORDER BY clause

Use the following steps to include a column in the ORDER BY clause.

**Procedure**

1. Add the desired column to the grid, as described in [Adding columns in the Grid panel](#).
2. Click the corresponding cell in the **Sort Order** column, and select the desired number (which represents the order in which that column is to be sorted).
   
   Initially, when no column is selected in the **Column** column, there is no number in the **Sort Order** drop-down list to indicate the sorting order. The numbers in the **Sort Order** drop-down list increase as you keep selecting the columns and specifying the sorting order of those columns.
3. Click a cell in the **Sort Type** column and specify the type of the sorting order: Ascending or Descending.
4. Click the **SQL** tab to view and verify the ORDER BY clause.

### Specifying the GROUP BY clause

Use the following steps to specify the GROUP BY clause.

**Procedure**

1. Add the desired column to the grid, as described in [Adding columns in the Grid panel](#).
2. Click the corresponding cell in the **Group By** column and choose one of these options from the drop-down list:
   - **Group By** — to include the corresponding column in the GROUP BY clause of the SQL.
   - **Having** — to include the corresponding column in the HAVING clause of the SQL.
   - **Expression** — to revert any of the previous selections.

### Specifying criteria on a column

Use the following steps to specify criteria on a column.

**Procedure**

1. Add the desired column to the grid, as described in [Adding columns in the Grid panel](#).
2. Specify the criteria on this column by entering them into the **Criteria** and the corresponding **Or** cells.
   
   Multiple criteria entered in one row (that is, horizontally) are combined using the OR logic; criteria entered in different rows (that is, vertically) are combined using the AND logic. A column does not have to be included in the result set to have criteria associated with it. For example, suppose you have an Order table that has a total_price column. If you want to retrieve large and small orders, you can enter ">100" (without the quotes) in the **Criteria** column, and "<10" (without the quotes) in the first **Or** column (assuming that is how you define large and small respectively).
Including functions in the SQL via the Grid panel

You can include a function through a Column cell or a Criteria cell. A function added through a Column cell is included in the SELECT clause of the SQL; a function added through a Criteria cell is included in the WHERE clause of the SQL.

Procedure

1. Add the desired column to the grid, as described in Adding columns in the Grid panel.
2. Right-click the desired cell in the Column section, and select: <function > <function type> > <function name>.
   Custom functions (administratively-promoted procedures that have a scalar output), may be available for inclusion from the SQL Grid in either columns or as criteria, if an administrator has promoted procedures with scalar outputs for general usage. Note: Administrators may promote procedures with a scalar output using the Administration > Custom Functions menu option.
3. In the Function Arguments Input window, specify the input argument(s) 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 clause.

Declaring variables in the SQL via the Grid panel

Follow these steps to declare variables in the SQL.

The variables defined through the following steps are considered virtual columns, and they are included in the SELECT clause of the SQL.

Procedure

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.
   • Specify the type of the variable in the drop-down list.
   • In the Default Value field, specify the default value for the variable.
3. Click OK.
   The value you defined should display in the Column cell in the format:

```sql
{DECLARE <variable_column_name> <data_type> DEFAULT <default_value>}
```

Example SQL: SELECT products.ProductID, {DECLARE columnName INTEGER DEFAULT 23} columnName, products.ProductName FROM /shared/examples/ds_inventory/products products WHERE columnName > products.ProductID

Designing a view's SQL in the SQL panel

You can use the SQL panel in the view editor to type and edit a view's SQL.

Note that the SQL for a view cannot contain an INSERT/UPDATE/DELETE statement. You have to include an INSERT/UPDATE/DELETE statement in a SQL script, which is described in the chapter Procedures in this guide.

Once you start typing in the SQL panel, the design in the Model panel becomes invalidated, so first save your current view with a different name if you want to use the Model panel later for the saved view, and use the SQL panel for the current view.
Also, you are responsible for the syntax of the SQL you type in the SQL panel, since IBM Cognos Virtual View Manager does not check the validity of the syntax.

**Note:** If any table/column names contain special characters, such as @, $, &, and so on, enclose those table/column names in double quotes (" ") in the query.

For details on the supported SQL features, see the chapter SQL Support in the Reference Manual.

**Procedure**

1. Click the **SQL** tab in the view editor.
2. Type the SQL for your view.
   - You can edit the SQL if it already exists. Also, using the **Insert from File** and **Save to File** buttons, you can upload an existing SQL statement or save the current SQL respectively.
3. Save the view.

**About generating a Model from the SQL**

You can generate a model from the SQL.

Clicking the **Generate Model** toolbar button in the view editor's SQL panel can regenerate a model from the SQL displayed in the SQL panel.

The model generator does not support all of IBM Cognos Virtual View Manager's SQL syntax.

Specific SQL features that are not supported by the model generator are as follows:

- UNION
- INTERSECT
- EXCEPT
- EXISTS
- scalar subqueries
- derived tables
- IN clause with a subquery
- quantified subquery
- INSERT operation
- UPDATE operation
- DELETE operation

Error messages will result for queries that include the above SQL features.

In the regenerated model, the tables would be joined at the top (joining the table tiles), not at the appropriate columns under the following circumstances:

- If the ON clause of the JOIN has a function; an OR condition; or an IN, LIKE, BETWEEN, or IS NULL predicate.
- If the join is a self-join (reflexive join); that is, no columns are involved or only columns from the same table are involved in the join.

The model generator accepts all INNER/OUTER JOINs. However, it may not be able to match the columns originating from the left and right sides of the join. When both columns cannot be identified, in the model generated in the **Model**
Designing column projections in the Column panel

Use the following steps to design column projections.

The columns are designed in the Column panel.

Procedure
1. Open the Columns panel in the View Editor.
2. Select Design Mode, click Add, and select a data type from the drop-down list.
   A default name and data type for the new column are displayed.
3. Optional: Delete the default name and type the column name to be included in the output.
4. Right-click the data type label and specify the proper data type for the column.
   Once all the columns are specified exactly in the order in which they need to be displayed in the result set, you can use them to do a top-down design of the SQL in the SQL panel. Note that the SQL in the SQL panel is not automatically updated to reflect the design created in the Columns panel, and vice versa.
   Therefore, edit the SQL so that the columns designed in the Columns panel are included in the SELECT statement.

Obtaining view details

Use the following steps to obtain details about a view.

Procedure
Click the Info tab in the view editor.
The Info panel displays the following details about the view:
• Name — user-provided name for the view
• Type — type of the resource; View
• Owner — name of the owner who created the view
• Owner Domain — domain to which the owner of the view belongs
• Annotation — notes about the view.
See also Annotate a resource

Executing a view

Executing a view means executing the view's SQL. Assuming that you have a view designed and saved, you can execute its SQL to see the result.

See "Tasks involved in designing views" on page 81 for details on designing a view.

This section describes how to execute a view's SQL from IBM Cognos Virtual View Manager.

For executing a view's SQL through JDBC/ODBC client programs, see Client Interfaces
Executing a view may be affected if the view uses a data source that was originally added to Virtual View Manager using the pass-through mode without saving the password. To execute such a view, you must log in with the same login credentials (user name and 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 DB2 Data Source.

Executing a view from Virtual View Manager would be affected if the view’s SQL is `SELECT *` on a view that includes a table with column-based security. See Privilege Enforcement on Sources Underlying a Resource.

**Procedure**

1. Double-click the desired view to open it, or right-click the view and select **Open**.
2. Click the **Execute** button on the toolbar of the editor.

The **Result** panel displays the view execution results in 50 rows at a time, as shown next. Use the **Next** button to view the next 50 rows.

---

### Rebinding a view

A view depends upon 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 in the following situations:

- When you decide to rebind the view to different sources, you can do that through the view’s editor.
- When there’s an error in executing a view because a source with which the view was initially bound doesn’t exist anymore and that you want to rebind the view with a new source.

**Procedure**

1. Open the view.
2. Click the **Show Rebind Panel** button on the toolbar.

   The **Rebind** panel opens in the lower part of the view’s editor and displays the dependencies as determined by the last successful save.
3. In the **Rebind** panel, select the desired source displayed in the left pane, and click the **Rebind** button.

   The **Rebind** window displays, showing the resources available for binding.
4. In the **Rebind** window, specify the desired source with which to rebind the view, click **OK**, and save the view.

   If you make any modification to the view or its dependencies, the binding is automatically updated when you save the view and the Rebind panel reflects the change accordingly. For example, if your view has a data source dependency named `ds_orders`, and you rename `ds_orders` to `ds_orders_renamed` and save the view, the **Rebind** panel would list the dependency as `ds_orders_renamed`.

   See [Seeing a view’s dependencies](#) for details on a view’s dependency.
Seeing a view’s dependencies

When you are working on a complex query, it is useful to know what resources are involved so you can better understand the query’s performance.

The Dependency panel shows a view’s SQL and dependencies.

Procedure
1. Right-click the desired view in the resource tree, and select Show Dependencies. Or, open the view and select the Show Dependency Panel toolbar button.

   The Dependency panel opens in the lower section of the view’s editor and displays all the resources underlying the view in a graphical format. In the graphical format, the view’s dependencies are shown on its left, whereas its dependents are shown on its right. In general, a given resource’s dependencies are shown on its left, whereas its dependents are shown on its right.

   To scale the size of the graph, use the input box which is on the right side of the Expand/Collapse Dependents button. Type a positive numeral in this box. The view's SQL is displayed on the right side of the Dependency panel.

   When you place the cursor on a given view, its projections are listed as shown in the next screen.

2. Click the Switch Graph/Tree View toolbar button, the display changes to a tree format.

   The top level node in the left pane of the Dependency panel represents the view, and its SQL is displayed on the right.

   Three folder-containers (Dependencies, Dependents, and Projections), representing the view’s dependencies, dependents, and projections, are subsumed under the top-level node.
Chapter 5. Procedures

This chapter describes procedures—what they are, the types that can be used, and how to use them.

This chapter assumes that you are familiar with the information presented in the preceding chapters.

About procedures

Procedures are pre-defined programs that can be executed to query and manipulate data stored in a data source. They have scalar input and/or output parameters. Some of them return one or more cursors to represent complex output data that is in tabular form.

IBM Cognos Virtual View Manager supports the following types of procedures:

- SQL scripts
- custom Java procedures
- transformations
- packaged queries
- parameterized queries
- physical stored procedures

Although different languages (Virtual View Manager SQL script, Java, SQL native to data sources, XSLT, XQuery) are used to formulate the different types of procedures, they all function alike in Virtual View Manager. You can use specific types of procedures for your specific needs. You can also invoke one procedure type from within another.

Note: If you want to make a procedure available to client programs through a JDBC/ODBC/SOAP connection, you must publish that procedure.

About creating procedures

You can use any text editor to create the text for a procedure, and add it to the server. You can also use the procedure editor in IBM Cognos Virtual View Manager to create and edit most of the procedures (see The Procedure Editor next). You cannot edit Java procedures in Virtual View Manager.

You can use any location except Virtual View Manager Data Services in the resource tree to add a procedure to the server.

The procedure editor

Virtual View Manager provides a procedure editor for creating and editing procedures. This editor opens automatically when you create a procedure. At any time later, you can open the editor for a procedure by double-clicking the procedure name in the resource tree.

This section introduces the Procedure editor panels and the Procedure editor toolbar available in the procedure editor.
**Procedure editor panels**

The procedure editor has specific panels for viewing and editing the various properties of a procedure, which are described next. You can open each of these panels by clicking the corresponding tab located in the lower section of the editor.

The panels that are available depend on the type of procedure:

<table>
<thead>
<tr>
<th>Procedure type</th>
<th>Available panels</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL scripts</td>
<td>SQL Script, Parameters, Caching, Info</td>
</tr>
<tr>
<td>Custom Java procedures</td>
<td>Parameters, Caching, Info</td>
</tr>
<tr>
<td>XQuery transformations</td>
<td>Model, XQuery, Inputs, Outputs, Caching, Info</td>
</tr>
<tr>
<td>Packaged queries</td>
<td>SQL, Parameters, Caching, Info</td>
</tr>
<tr>
<td>Parameterized queries</td>
<td>Model, Grid, SQL Script, Parameters, Caching, Info</td>
</tr>
<tr>
<td>Physical stored procedures</td>
<td>Parameters, Caching, Info</td>
</tr>
</tbody>
</table>

**Model panel (procedure editor)**

The Model panel is available for Parameterized Queries and XQuery Transformations.

In the case of a parameterized query, use the Model panel for adding resources to a parameterized query when designing the query. You can add any type of table (relational, LDAP, delimited file) and a procedure that outputs either a set of scalars or exactly one cursor. This panel works in conjunction with the Grid (Procedure Editor) panel where you can specify the output column projections as well as the constraints on a query. This panel is permanently disabled if you edit the script in the SQL Script panel for the parameterized query.

In the case of an XQuery transformation, use this panel for specifying the sources and target values that provide the data for the output XML document. This panel is permanently disabled if you edit the script in the XQuery panel.

**Grid panel (procedure editor)**

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 in conjunction with the Model (Procedure Editor) panel. This 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 highlighting 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 in conjunction with the Parameters panel, in the sense that the parameters
you define in the SQL Script panel must exactly 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 parametrized query.

**SQL panel (procedure editor)**

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 in conjunction with the Parameters panel, in the sense that the parameters you use in this panel must exactly match the parameters designed in the Parameters panel, including the order in which they are provided.

**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 in conjunction with the SQL Script panel (or, the SQL panel as the case may be) in the sense that the parameters you design in this panel must exactly match the parameters defined in the SQL Script panel (or, the SQL panel as the case may be) including the order in which they are provided.

In the case of Java procedures and transformations, you can only view the parameters in the Parameters panel, since the parameters are defined in the source Java code (in the case of a Java procedure) and in the XML or WSDL (in the case of a transformation). Each parameter is displayed with its 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.

In the case of an SQL script, a packaged query, or a parametrized query, use the Parameters panel to design and edit the parameters.

- The Design Mode check box lets you formulate the input and output parameters for a procedure. Formulating all the parameters using the design mode lets you do a top to bottom design in the panel where the SQL is created (in the SQL Script panel or the SQL panel, as the case may be). When you formulate the parameters in the Parameters panel, the SQL implementation is not automatically updated. You must make sure that the parameters used in the SQL and those that are designed in the Parameters panel using the design mode exactly match; otherwise, the procedure will throw an error when executed.
- The Add button is enabled when the Design Mode option is selected in the Parameters panel. It lets you add parameters when you design an SQL script, a packaged query, or a parameterized query.

**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.

**XSLT panel**

The **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 once you start editing the XSLT, the mapping in the **Data Map** panel is permanently disabled.

**XQuery panel**

The **XQuery** panel is available for XQuery transformations. Use this panel to view the XQuery that would return an XML document when the transformation is executed. The system auto-generates the XQuery when you use the **Model** (Procedure Editor) panel to design the output for the transformation.

You can edit the XQuery in this panel, but once you start editing the XQuery, the **Model** panel is permanently disabled.

**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 and consequently the Data Map panel is permanently disabled, 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 and consequently the Model panel is permanently disabled, 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.

**XSLT debug input panel**

The **XSLT Debug Input** panel is available for XSLT transformations, and opens 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 when a transformation is executed. Use this panel to view the output from the XSLT. The result columns are displayed with data.

**Caching panel (procedure editor)**

The Caching panel lets you create a cache for the procedure.

**Info panel (procedure editor)**

The Info panel is available for all types of procedures. This panel displays the following details about a procedure:

- **Name**—user-provided name and location
- **Type**—resource type
- **Owner**—owner name
- **Owner Domain**—domain to which the owner of the view belongs
- **Source**—source of a transformation
- **Annotation**—annotation on the resource

For XQuery transformations, the Info panel also provides:

- **Target Definition Set**—the path to the definition set containing the schema
- **Target Schema**—the fully qualified namespace for the schema

Only the annotation can be edited by a user with the WRITE privilege on the procedure.

The Transaction Options section in the Info panel has a check box labeled **Execute only once per transaction for each unique set of input values** which you can select if you want to ensure the consistency of procedure execution results during a transaction.

By selecting this check box, you ensure that the execution results of a procedure are kept intact for the entire scope of a transaction as long as the procedure's input parameter values remain the same.

**Rebind panel (procedure editor)**

The Rebind panel is available for all types of procedures, and opens when the Show Rebind Panel button ( ) is clicked. Use the Rebind panel to view the schema of the resources that a procedure depends upon, and also to rebind a procedure to a different resource.

**Result panel (procedure editor)**

The Result panel is available for all types of procedures, and opens when the Execute button ( ) is clicked to execute the procedure. Use this panel to view the output results of an executed procedure.

The Cancel button displays in the Result panel during query execution. You can use the Cancel button to cancel the in-progress execution.
Also, if you close the Result panel, any in-progress procedure execution is cancelled.

**Procedure editor toolbar**
The following table lists the procedure editor’s toolbar buttons (in alphabetical order) and their functionality.

<table>
<thead>
<tr>
<th>Button</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Choose Source" /></td>
<td>Choose Source</td>
<td>Opens a window for selecting a source for the corresponding target selected.</td>
</tr>
<tr>
<td><img src="image" alt="Choose Target Schema" /></td>
<td>Choose Target Schema</td>
<td>Lets you select an XML target schema.</td>
</tr>
<tr>
<td><img src="image" alt="Create Link" /></td>
<td>Create Link</td>
<td>Creates a link between the specified source and the target.</td>
</tr>
<tr>
<td><img src="image" alt="Create Link And Target" /></td>
<td>Create Link And Target</td>
<td>Creates a target for the specified source and links them.</td>
</tr>
<tr>
<td><img src="image" alt="Cycle I/O Direction" /></td>
<td>Cycle I/O Direction</td>
<td>Cycles through input/output directions. Blue arrow indicates an input parameter. Red arrow indicates an output parameter.</td>
</tr>
<tr>
<td><img src="image" alt="Delete" /></td>
<td>Delete</td>
<td>Removes a resource from the tree and the repository.</td>
</tr>
<tr>
<td><img src="image" alt="Design By Example" /></td>
<td>Design By Example</td>
<td>Opens the Design By Example window for defining parameters.</td>
</tr>
<tr>
<td><img src="image" alt="Execute" /></td>
<td>Execute</td>
<td>Executes a procedure.</td>
</tr>
<tr>
<td><img src="image" alt="Format Query" /></td>
<td>Format Query</td>
<td>Formats the script.</td>
</tr>
<tr>
<td><img src="image" alt="Insert from File" /></td>
<td>Insert from File</td>
<td>Opens a window for specifying a location from which to upload the SQL or SQL script for a procedure.</td>
</tr>
<tr>
<td><img src="image" alt="Move Parameter Down" /></td>
<td>Move Parameter Down</td>
<td>Moves a parameter down.</td>
</tr>
<tr>
<td><img src="image" alt="Move Parameter In" /></td>
<td>Move Parameter In</td>
<td>Moves a parameter inward.</td>
</tr>
<tr>
<td><img src="image" alt="Move Parameter Out" /></td>
<td>Move Parameter Out</td>
<td>Moves a parameter outward.</td>
</tr>
<tr>
<td><img src="image" alt="Move Parameter Up" /></td>
<td>Move Parameter Up</td>
<td>Moves a parameter up.</td>
</tr>
<tr>
<td><img src="image" alt="Navigate" /></td>
<td>Navigate</td>
<td>Opens the Navigator window.</td>
</tr>
<tr>
<td><img src="image" alt="Parse" /></td>
<td>Parse</td>
<td>Parses the XML schema.</td>
</tr>
<tr>
<td><img src="image" alt="Rebind" /></td>
<td>Rebind</td>
<td>Rebinds a procedure (except Java procedure) to a different resource.</td>
</tr>
</tbody>
</table>
This section describes how to create a SQL script. SQL scripts are procedures written in IBM Cognos Virtual View Manager's SQL script language.

### Creating SQL scripts

Follow these steps to create an SQL script.

**Procedure**

1. Right-click the container to which you will 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 right pane displaying the **SQL Script** panel.

   Use the **SQL Script** panel to formulate and edit the SQL script. The parameters you define in the **SQL Script** panel must exactly match the parameters that already exist in the **Parameters** panel, including the order in which they are defined.

   The **SQL Script** panel has syntax highlighting feature, which can be toggled with the **Toggle Syntax Highlighting** button ( ). The highlighting applies to all the keywords in the SQL script language.

3. If you have a script stored in your file system, upload it using the **Insert from File** button ( ) on the editor's toolbar, or type the script between the lines **BEGIN** and **END**.
Keep these rules in mind:

- You can comment a block of text by preceding every line of the comment with two dashes (--) or by enclosing the comments within /* and */.
  
  For example,
  
  ```
  /*
  This
  is
  my
coment.*/
  ```

- You can use keywords in a SQL Script as long as you enclose them within double quotes.
  
  For example,
  
  ```
  SELECT
  "begin" INTO
  ```

- 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 want to do an INSERT/UPDATE/DELETE operation, you must include the INSERT/UPDATE/DELETE statement in an SQL script.

  For more information on the syntax of the SQL Script language, see the "SQL Script" chapter in the IBM Cognos Virtual View Manager Reference Guide.

4. Save the script.

Designing the parameters for SQL scripts

Follow these steps to design the parameters for an SQL script in the Parameters panel.

**Procedure**

1. Open the Parameters panel in the procedure editor, and select the Design Mode check box located at the top right.

   The Parameters panel becomes editable when you select Design Mode.

   **Note:** If you want to create a script by first designing its parameters, you must do so in the Parameters panel and then write a procedure in the SQL Script panel.

2. Click Add to start adding parameters, and select a 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.

   When you click Add and select a data type, a new parameter is added with a default name and the specified data type.

3. Name the parameter using one of these methods:
   - Type a name for the parameter next to the arrow, and click Enter.
   - Right-click the parameter name, choose Rename, and type the new name.

4. Optional: Change a data type
Right-click the data type name, and select **Change Type**.
Click the field displaying the data type name, and select the desired data type from the drop-down list.
Optional: Specify the length or the number of digits for certain data types.
Click **OK**.

5. Select the parameter, and click the **Cycle I/O Direction** button ( ) to indicate whether the parameter is input, output, or input/output.

- denotes an input parameter.
- denotes an output parameter.
- denotes an input/output parameter.

6. Use the navigation buttons to move a parameter up, down, left, or right.
The up/down triangle buttons are enabled when there is more than one parameter. The left/right triangles are enabled for a CURSOR parameter.

7. After specifying the desired number of parameters, save the edits.

8. In the **SQL Script** panel, complete the procedure using the parameters designed in the **Parameters** panel.

9. Save the script.

**Results**

Once the parameters in the design (**Parameters** panel) match the parameters in the script (**SQL Script** panel), the name of the script will be rendered normally in black.

**Note:** If a SQL script has an array as an input parameter, you can supply the input value when executing the script.

**Promotion of procedures to custom functions**

Procedures with a scalar output may be used to directly manipulate data as part of the SELECT or as criteria in a condition by promotion of the needed procedure for use as a Custom Function.

Administrators may promote SQL scripts and other procedures with a scalar output and zero to many inputs using the **Administration > Custom Functions** menu option.

**Java procedures**

Java procedures are programs that you can write in Java to access and use IBM Cognos Virtual View Manager resources.

You can add these procedures to Virtual View Manager as you would any other type of data source, execute them on a data source, use them in views/procedures, and publish them as Virtual View Manager database tables or Web services.

For details on the Virtual View Manager APIs for custom Java procedures and examples of Java procedures, see the “SQL” chapter of the *IBM Cognos Virtual View Manager Reference Guide*. 
**Viewing Java procedure parameters**

When you add a Java procedure to the server, the procedure name is displayed in the Virtual View Manager resource tree.

**Procedure**

Open the desired Java procedure in the resource tree.

The editor for the procedure opens on the right, displaying the Parameters panel. This panel displays the input/output parameters for the procedure. You can only view the parameters in the Parameters panel, since the parameters are defined/edited in the source Java code. Each parameter is displayed with its JDBC data type and the data type native to the corresponding data source, as shown in the next screen. The output parameters shown in this panel are rendered as columns in the result set when you execute the procedure.

---

**Transformations**

This section assumes that you have a firm understanding of XML, XQuery, and WSDL.

Since the data for modeling come from disparate sources including relational (tabular) and hierarchical (such as XML), IBM Cognos Virtual View Manager provides mechanisms to transform XML data into tabular format as well as to transform tabular data into XML format.

This section describes how to create the Virtual View Manager resource called transformation to transform data into a desired format.

**Transformation types**

Virtual View Manager has a built-in transformation mechanism which you can use to create the different transformations.

**Basic 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. Essentially, the XML document is flattened into a table that represents the document's data and nodes which are described in the following table:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INTEGER</td>
<td>unique identifier for an element or attribute value</td>
</tr>
<tr>
<td>parent_id</td>
<td>INTEGER</td>
<td>unique identifier for the parent of this element</td>
</tr>
<tr>
<td>depth</td>
<td>INTEGER</td>
<td>number of elements in path</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>local name of the current element or attribute</td>
</tr>
<tr>
<td>xpath</td>
<td>VARCHAR</td>
<td>Xpath expression that fully describes the location of the element or attribute</td>
</tr>
<tr>
<td>path</td>
<td>VARCHAR</td>
<td>path to the element or attribute. The path consists of a forward-slash separated list of enclosing element names ending with name</td>
</tr>
<tr>
<td>position</td>
<td>INTEGER</td>
<td>position of this element relative to the path attribute</td>
</tr>
</tbody>
</table>
### XSLT transformation

The XSLT transformation lets you define how the XML data should be transformed using a graphical editor in Virtual View Manager.

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:

```xml
<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 Virtual View Manager.

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 queries various sources and returns the result in an XML document.

An XQuery transformation is a Virtual View Manager resource that reads data from related tabular sources, such as tables, views, and procedures, and assembles the related data into an XML document that conforms to a user-specified schema which is called the target schema. An XQuery transformation looks and functions like any other procedure in the 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/edited in the definition set editor.

Virtual View Manager also provides a graphical editor for mapping the target schema and the tabular sources that will supply the data for the XQuery in the transformation.

### About creating XML, XSLT, or Streaming Transformations

You must follow three steps to create a non-XQuery (XML to tabular, XSLT, and streaming) transformation.

The steps are:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>VARCHAR</td>
<td>value of the element or attribute</td>
</tr>
</tbody>
</table>
• Create the transformation.
• Map the source data to the target output columns (XSLT and streaming only).
  The Data Map panel is available for XSLT transformations and streaming transformations. This panel lets you map elements from the source XML document to the target output columns.
• Add target output columns if desired (XSLT only).
  When you add target output columns via the Outputs panel for XSLT transformations, the XSLT in the XSLT panel is not automatically updated. You must manually update the XSLT.
  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 gets generated.

This section describes each of these procedures.

Creating non-XQuery transformations
Follow these steps to create a non-XQuery transformation.

Procedure
1. Have a source (XML or WSDL) for the transformation ready, and know its location on the server.
2. From an appropriate location in the resource tree, right-click and select New Transformation.
   Alternatively, at an appropriate location, you can select File > New > Transformation.
3. In the Create Transformation window, specify the desired type of non-XQuery transformation, and click Next.
   The following steps describe how to create an XSLT type of transformation.
4. Locate and select the source for the transformation in the displayed tree.
5. Type a name for the transformation in the Transformation Name field, and 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 Basic 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.
   The procedure to map source data to target output columns is described next.

Mapping source XML data to target output columns
Follow these steps to map source XML data to target output columns in the Data Map panel.

Procedure
1. Double-click the desired transformation in the resource tree.
   The transformation editor opens the Data Map panel.
At any time later, you can open the Data Map panel by clicking the Data Map tab in the editor.

2. Create target columns and map them to sources, using either the manual mapping or auto-mapping process below:

To create target columns and manually map them to sources:
- Select the outputs node (or an existing column) in the Target column, click Add in the editor's toolbar, and specify the data type for the output column in the drop-down list.
- An output column with a default name is created. You can create as many target columns as you need.
- Connect a source and a target by selecting them individually and clicking the Create Link button.
- You can map a single source to multiple targets, but you cannot map a single target to multiple sources. The data types of the source and the target should match. If there's no exact match of data types, you can match data types if one is castable to the other.

To automatically create target columns and links from a given source:
- Select the desired source in the Source column.
- 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 will create a separate target and a link for each source. Clicking Create Link And Target several times will create that many targets and links pointing to the same source.

3. To rename a column and change its data type:
- Right-click the column name, select Rename, and supply a new name.
This name will be displayed as a column name in the output when the transformation is executed.
- Right-click the column name, select Change Type, and specify a new type.
The data types of the source and the target should match. If there's no exact match of data types, you can match data types if one is castable to the other.

4. To unlink a target and a source, grab and drag the link away from the target/source, and release the mouse button.

5. To align a target with its source, move the target up or down using the up (△) or down (▼) buttons.

6. Save your edits.

Note: When you map source elements to target output columns in the Data Map panel, the XSLT is automatically generated in the XSLT panel.

Adding target output columns
Follow these steps to add target output columns via the XSLT and Output panels.

Procedure
1. Double-click the desired transformation in the resource tree to open the editor.
2. Edit the XSLT in the XSLT panel, as needed.
The Data Map panel is permanently disabled, but the Outputs panel becomes editable and the Add button in the toolbar is enabled.
Add target output columns as you would design parameters for an SQL script in the Parameters panel.

3. Update the XSLT in the XSLT panel, and save the transformation.

**XQuery transformations**

The transformation described in this section is based on a pre-created XML definition set (InventoryTransactions) which you can find in the resource tree in /Shared/examples.

**Creating XQuery transformations**

Follow these steps to create an XQuery transformation.

**Procedure**

1. Have an XML definition set ready. The definition set will provide the target schema for the output document.
2. From an appropriate location in the Virtual View Manager resource tree, right-click and select New Transformation, or select File > New > Transformation.
3. In the Create Transformation window, select XQuery Transformation, and click Next.
4. Locate and select the desired XML definition set in the resource tree displayed in the left panel.
5. In the right panel, where all the elements in the XML definition set are displayed:
   - Optional: Expand the nodes and view the elements to see what is available.
   - Select a top level element.
     This top-level element will determine the type and name of the root element in the output XML document.
     In the current example, there is only one top-level element (InventoryTransactions).
     If you use an XML definition set similar to UserSchema, which you can find in /Virtual View Manager Data Services/Web Services/system/admin/user/userPort/.
     You will have many top-level elements for selection, as shown in the next screen.
6. After selecting the desired top-level element, supply a name for the transformation in the Transformation Name field, and click Finish.
   The transformation is added to the resource tree and its editor opens in the right pane displaying the Model panel by default.
7. View the Model panel.
   The Model panel in the XQuery transformation is similar to the Data Map panel in the XSLT transformation editor but has the opposite functionality. It lets you map source data to the target schema and transforms tabular data into XML data.
   The Model panel in the XQuery transformation has the following columns—Target, Target Value, Source, Source Alias, and Source Settings—which are described next.
   The following screen contains sample entries in all the columns in the Model panel of the XQuery transformation. After reading through the descriptions of
these columns, proceed with the section “Specifying settings for target sources” on page 106 to complete the definition of this transformation.

- The **Target** column

The **Target** column displays the XML schema that is uploaded (via 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 (Transaction\[\] in the previous screen). Elements rendered in italicized type are optional.

Run the cursor over the entries in the **Target** column to view the tool tip for each entry. Press the Esc key to dismiss the tool tip.

- The **Source** column

The **Source** column lets you specify the tabular sources that provide data for the resulting XML document. To specify a source in this column, double-click the cell corresponding to the desired top-level element in the **Target** column; or, select the desired top-level element in the **Target** column and click the **Choose Source** toolbar button ( ). In the window that opens, select the desired source. 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), as well as the input to the XQuery. The scope defines what resources are available to the value expression (see The **Target Value** Column).

In the following screen, source scopes are marked by connecting a target to a source in its scope. Since the global input (**SupplierName**) is in the scope of all the targets, its connection to a specific target is not marked.

The scope for **Transaction** is its own source, the inventorytransactions table, and the global input SupplierName (line 1).

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, as well as 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). Note that the sources (purchaseorders and suppliers) in the scope of its sibling/peer resource (PurchaseOrder) are excluded from the scope of **Product**.

To view the source scope for an element, click a cell in the **Target Value** column and view the drop-down list, as shown in the next screen. The global input parameter is listed at the end. If there are more than one input parameter, they would be listed in the order in which they are defined in the **Inputs** panel.

- The **Target Value** column
The **Target Value** column contains the actual expressions (projections) that determine the values for the leaf-node elements in the **Target** column. The expression (projection) in each cell is relative to the views, tables, and procedures that are currently in scope (see **Source** in the preceding paragraph for a description of scope). The **Target Value** column is similar in functionality to the column named **Column** in the view editor's **Grid**.

To specify a target value, click the cell corresponding to the desired target in the **Target** column, and select a value from the drop-down list.

**Note:** Only the alias of a source, not the actual name of the source, is used in the **Target Value** column.

- **The Source Alias column**

  The **Source Alias** column is for specifying aliases for the sources specified in the **Source** column. The aliases are used when join conditions, filters, and inputs are supplied for a source value. The **Source Alias** column is similar to the **Alias** column in the view editor's **Grid**. 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**

  The **Source Settings** column 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. When clicked, the Schema icon ( ) displays the schema for the corresponding source.

  Source settings are specified in the lower section of the editor, which you can open by clicking the **Show/Hide Source Settings** toolbar icon ( ) or by clicking an icon in the **Source Settings** column. The settings for a source can be a join XPath expression, filter XPath expression, sort order, or an input parameter as in a procedure, as shown in the next screen. Only the alias for a source, not the actual source name, is used in a value expression.

  In the **Join** panel you can specify a join relation between the current source and one of its direct ancestors. For example, `PurchaseOrderID = $INV/PurchaseOrderID` which means that `PurchaseOrderID` in `PurchaseOrder` is joined to `PurchaseOrderID` in the ancestor `InventoryTransactions`.

  In the **Filter** panel you can specify a filter on the current source (the equivalent of a **WHERE** clause in a **SELECT** statement). For example, `$INV/TransactionID > 25` is a filter on `TransactionID` in the Source named `inventorytransactions`. The filter is similar to an entry in the **Criteria** column in the view editor's Grid. The filter is used to constrain the data returned from this resource.

  In the **Sort Order** panel you can 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 **Inputs** panel, you specify the values for a source's inputs if the source is a procedure with input parameters.

  Once you specify a setting, the corresponding icon is shown in the **Source Settings** column. You can click this icon at any time to view/edit the setting.

**Specifying settings for target sources**

Follow these steps to specify the sources, values, and other settings for target sources in an XQuery transformation.
Procedure
1. To specify a source, double-click the cell corresponding to the desired top-level element in the Target column; or, select the desired top-level element in the Target column and click the Choose Source toolbar button ( ). In the window that opens, select the desired source, and click OK.
2. In the Source Alias column, you can supply an alias for the source you just added.
   The aliases are used when join conditions, filters, and inputs are supplied for a source value.
3. To specify a value for a target, click the Target Value cell corresponding to the desired target in the Target column, and select a value from the drop-down list.
   Note: For a target value, you can supply a literal value or a system function that would be evaluated at runtime. Literal values must be enclosed in single/double quotes.
4. To specify a source setting, first click the desired icon in the Source Settings column or click the Show/Hide Source Settings toolbar icon ( ) to display the source settings panel in the lower section. Then, open the desired panel—Join, Filter, Sort Order, or Inputs—and supply the settings.
   For example, 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 $INV/PurchaseOrderID contains a reference to the alias (INV) for the parent source inventorytransactions preceded by the dollar sign ($).
5. 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).
   • Click the Inputs tab to open the Inputs panel.
   • Click the Inputs icon in the Source Settings column.
     In the Inputs panel, the name(s) of the input(s) are displayed, and the Is NULL check box is selected by default.
   • 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
Follow these steps to specify global input parameters in an XQuery transformation.

Procedure
1. Click the Inputs tab (in the upper section of the editor).
2. Click the Add button 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. You can click the Outputs tab to view the outputs.
4. Save the transformation.

Viewing and editing XQueries
Follow the step below to view and edit an XQuery.
Procedure

After specifying the sources, values, and other settings for target sources as needed, click the XQuery tab in the editor to view the auto-generated XQuery for the model designed in the Model panel.
You can manually edit the XQuery text in this panel, but be aware that such editing will permanently disable the Model panel for the current transformation.

Viewing the output parameters

Follow the step below to view the output parameters in an XQuery transformation.

Procedure

After specifying the sources, values, and other settings for target sources as needed, click 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.

Packaged queries

Packaged queries let you use database-specific queries within the IBM Cognos Virtual View Manager.

Sometimes, you may already have a complex query written for a particular database and it may not be feasible or desirable to rewrite that query in Virtual View Manager’s SQL. For example, your query may require a database-specific feature not available in Virtual View Manager or perhaps the query takes advantage of the database-specific feature for performance reasons. In such cases, your database-specific query can be employed as a packaged query and subsequently be used in other queries.

Every packaged query is associated with a specific data source.

A packaged query is stored in Virtual View Manager 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 OUT parameter that is a CURSOR with at least one column.

Since Virtual View Manager cannot automatically determine the required inputs and outputs of a database-specific query, it is necessary for the user to supply this information.

Creating packaged queries

Follow these steps to create a packaged query.

Procedure

1. Right-click an appropriate location in the resource tree to add the packaged query, and select New Packaged Query.
   The Add Packaged Query window opens.
2. Specify 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.
3. Type a name for the packaged query in the Name field, and 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 exactly match the parameters designed in the Parameters panel, including the order in which they are provided.

4. In the SQL panel for the packaged query:
   • If the database-specific query already exists in the file system, upload the file using the Insert from file button ( ) in the editor’s toolbar. The SQL from the file will be copied onto the SQL panel.
   • If the database-specific query is not available elsewhere, type it in the SQL panel.
   In the SQL, you don’t have to supply the full path to the data source since the packaged query is already associated with the data source. Also, the packaged query must have exactly one output parameter that is a CURSOR with at least one column. In the above screen, the data source is order details (specified in the FROM clause).
   Notice the curly braces and numerals used to define the input parameters (specified in the WHERE clause).

5. Click the Parameters panel in the editor.
   Use the Parameters panel to design the input and output parameters for the query. The parameters designed in the Parameters panel must exactly match the parameters defined in the SQL panel. Once 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 this example, add one output parameter (UnitPrice) and two input parameters (ProductID and Status) in the Parameters panel.
   • Select the output parameter named result, click Add, and select Decimal > DECIMAL as the data type.
     A new parameter is added with the specified data type.
   • Rename the newly added parameter as UnitPrice (to indicate the output), and press Enter.
     Make sure that this parameter is a part of the CURSOR output, by moving it to the right of result. Use the ( ) button, if necessary.
   • Add another parameter of the type Integer > INTEGER.
     This is the first input parameter in our example. If it’s 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, and you’ll see an arrow icon next to it.
     The right arrow denotes an input parameter, and the left arrow an output parameter.
   • Rename the parameter as ProductID, and press Enter.
   • Add another input parameter named Status of the type String > CHAR.

6. Save the packaged query.
   Executing a packaged query is similar to executing a parameterized query.

**Input parameters for packaged queries**

An input parameter is not required for a database-specific query. However, when creating a packaged query you can define the database-specific query to require one or more values in order to execute.

For each input value in the database-specific query, you must create an input parameter in the packaged query using the editor in IBM Cognos Virtual View Manager. 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, which you can define in the packaged query. When the server executes the packaged query, it will insert real 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**

Package query supports the following input data types:

- BINARY
- VARBINARY
- DECIMAL
- DOUBLE
- FLOAT
- NUMERIC
- BIGINT
- BIT
- INTEGER
- SMALLINT
- TINYINT
- CHAR
- LONGVARCHAR
- VARCHAR
- DATE
- TIME
- TIMESTAMP
- CURSOR
- XML

**Input substitution pattern**

Packaged query lets you use arbitrary input substitution patterns. If there is no input defined, the package query doesn’t need any substitution pattern. However, if the package query defines N inputs, it will contain {0}...{N-1} substitution patterns, where the numbers between {0} and {N-1} are continuous.

When you use STRING and DATE data types, you should use a single quote before and after the input substitution pattern. This rule doesn’t apply to numeric data types.

See the examples given next.
Invalid queries

The following queries are invalid:

- **Invalid Query A**

```sql
SELECT customer.name, customer.balance FROM customer
WHERE customer.id = {0} AND customer.status = {2}
```

What is wrong here?

Note that an input with the substitution pattern `{1}` is missing in this query.

- **Invalid Query B**

```sql
SELECT customer.name, customer.balance FROM customer
WHERE customer.id = {0} AND customer.zip = '{1}' AND customer.status = {2} AND customer.email = {3}
```

What is wrong here?

The database-specific query defines 3 inputs but the packaged query has 4 input substitutions. `{3}` makes the query invalid. You need to define one more input.

How are the input parameters evaluated?

This section illustrates how the packaged query input parameters are evaluated at runtime.

The following queries are valid:

- **Valid Query 1**

```sql
In the following example, `{0}` is a string and `{1}` is a number:

```sql
SELECT customer.balance FROM customer
WHERE customer.name = '{0}' AND customer.id = {1}
```

Note that the pattern for a string is enclosed in single quotes.

In this example, the first input value will replace all the occurrences of `{0}`, and the second input value will replace `{1}`, and so on.

- **Valid Query 2**

Each input substitution pattern is allowed to appear more than once in a packaged query, as in the following example:

```sql
SELECT customer.name, customer.balance FROM customer
WHERE (customer.id = {0} AND customer.zip = '{2}')
OR (customer.id = {1} AND customer.zip = '{2}')
```

Suppose, if the first input value is 101, the second is 102, and the third is CA94403, then the substitution occurs as follows:

Packaged query before substitution

```sql
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

```sql
SELECT customer.name, customer.balance FROM customer
WHERE (customer.id = {0} AND customer.zip = '{2}')
OR (customer.id = {1} AND customer.zip = '{2}')
```
WHERE
(customer.id = 101 AND customer.zip = 'CA94403')
OR (customer.id = 102 AND customer.zip = 'CA94403')

• Valid Query 3
If {i} is a pattern that needs to be preserved instead of replaced, then '{' and '}'
need to be escaped with the escape character \ (backward slash).

Packaged query before substitution

SELECT customer.name, customer.balance
FROM customer WHERE
(customer.id = {0} AND (customer.zip = 'preserved {1} input')
OR (customer.id = {1} AND customer.zip = '{2}')

Packaged query after substitution

SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = 101
AND (customer.zip = 'preserved {1} input')
OR (customer.id = 102 AND customer.zip = 'CA94403')

Note, the back slash (\) is removed from the original query.

• Valid Query 4
If '\{i\}' is a valid string that cannot be altered, add one more escape character,
such as '\\{i\}'.

Packaged query before substitution

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

SELECT customer.name, customer.balance
FROM customer
WHERE
(customer.id = 101
AND (customer.zip = '\\{2\}')
OR (customer.id = 102 AND customer.zip = 'CA94403')

• Valid Query 5
If a query input is a string containing a single quote, the single quote will be
escaped by the packaged query to become two single quotes after substitution.

Packaged query before substitution

SELECT customer.balance
FROM customer
WHERE customer.name = '{0}' AND customer.id = {1}

Packaged query after substitution

The first input is "Michael's son" and the second one is "123"

SELECT customer.balance
FROM customer
WHERE customer.name = 'Michael''s son' AND customer.id = 123

• Valid Query 6
If a query input contains a number enclosed in curly braces as in '{55}', you
should escape each curly brace with a backslash. If the query contains a string
enclosed in curly braces, as in '{the Second}', you don't need to escape the curly
braces.

Packaged query before substitution

SELECT customer.balance
FROM customer
WHERE customer.name = 'George {the Second}' AND customer.id = '{\{55\}}'

Packaged query after substitution
SELECT customer.balance
FROM customer
WHERE
  customer.name='George (the Second)'
  AND customer.id='[55]'

Parameterized queries

IBM Cognos Virtual View Manager provides support for creating SQL SELECT statements in which you can include input parameters. This feature is called a parameterized query.

A parameterized query is essentially a SQL SELECT statement that can have named parameters in the projections or selections, and it is implemented as a single-statement SQL script. The SQL script code is automatically generated when you design a parameterized query with the Model and Grid panels. For details on the language of SQL script, see the "SQL Script" chapter of the IBM Cognos Virtual View Manager Reference Guide.

The resources you can use in a parameterized query are tabular data and any type of procedure that outputs either a set of scalars or exactly one cursor. A parameterized query lets you include {param NAME TYPE} structures in the Grid panel to parameterize parts of the view that is being built. The resource itself is actually an SQL script with a model, so it functions like a procedure within the system.

Creating parameterized queries

Here are some quick steps for creating a parameterized query. These steps are for using a table in a parameterized query. Details of adding parameters and diagrams are given later in the chapter.

Procedure

1. Select File > New > Parameterized Query from an appropriate location in the resource tree, and give it a name.
   Alternatively, you can right-click at the appropriate location in the resource tree, and select New Parameterized Query.
2. Add the desired table to the Model panel in the editor that opens on the right.
3. Click the Grid tab to open the Grid panel.
4. Specify a column to project, or leave the cell entry as * (to select all the columns).
5. Right-click a blank column cell, and select Parameter.
6. In the Add Parameter dialog:
   • Type a unique name for the parameter in the Parameter Name field.
   • Specify the data type of the parameter in the Data Type drop-down list, and click OK.
     This entry is displayed in the Grid in the following format:
     {param <parameter name> <parameter data type>}
     The corresponding Alias cell is automatically filled with a unique value.
When you execute the parameterized query, the Enter Values for Input Parameters window opens prompting you to provide a value for the parameter you added.
Details of adding input parameters

You can define input parameters in the SELECT clause or the WHERE clause or the FROM clause of a parameterized query.

- To include an input parameter in the SELECT clause, you must add it from a Column cell in the Grid panel.
  For more information, see "Including parameters in the SELECT clause of parameterized queries."

- To include an input parameter in the WHERE clause, you must add it from a Criteria cell in the Grid panel.
  For more information, see "Including parameters in the WHERE clause of parameterized queries" on page 115.

- To include an input parameter in the FROM clause, you should include a procedure with an input parameter in the parameterized query.
  For more information, see "Including parameters in the FROM clause of parameterized queries" on page 115.

About parameter names

When you type a parameter, it must be of the format {param <name> <type>}. The parameter name is case-sensitive, so the case (upper/lower) you use in the parameter name is recognized by the system.

Several parameters can have the same name, provided they are distinguished by lower/upper case letters and data types.

Several parameters can have the same name with the same data type, provided they are distinguished by different aliases.

Including parameters in the SELECT clause of parameterized queries:

Follow these steps to include a parameter in the SELECT clause of a parameterized query.

Procedure

1. In the resource tree, right-click an appropriate location 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 opens in the right pane displaying the Model panel.
   Use the Model panel for adding resources to a parameterized query when designing the query. You can add any type of table (relational, LDAP, delimited file) and a procedure that either contains scalar outputs or only one cursor output.
3. Add the desired resources to the Model panel.
   For example, add the products table from the ds_inventory data source in /Shared/examples.
4. Click the Grid tab.
   The Grid panel in this editor is functionally similar to the Grid panel in the View editor.
5. In the Grid panel:
   - Click the first Column cell, and select the UnitPrice column.
• Right-click the next Column cell, and select **Parameter**.
• In the **Add Parameter** dialog:
  Type myParameter in the **Parameter Name** field, and select DECIMAL as its
data type. This entry displays as `{param myParameter DECIMAL(32,2)}` in the **Column**
cell.
  Change the automatic entry (Expr1 or something similar) in the **Alias** cell to
  specificPrice.

6. 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 we defined in Step 5 (myParameter) appears as the 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.

7. Click the **Parameters** tab to view how the parameters are displayed.
The **Parameters** panel here is functionally similar to the **Parameters** panel in
the editor for an SQL script.

8. Save the parameterized query.

**Including parameters in the WHERE clause of parameterized queries:**

Follow these steps to include a parameter in the WHERE clause of a parameterized
query.

**Procedure**
1. In the resource tree, right-click an appropriate 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. In the **Grid** panel:
   • Click the first **Column** cell, and select ProductName.
   • Click the next **Column** cell, and select ProductID.
   • Right-click the **Criteria** cell corresponding to `product.ProductID` under
     **Column**, and select **Parameter**.
   • In the **Add Parameter** window, supply a name for the parameter (say id),
     select the data type INTEGER (since ProductID is of INTEGER type), and
     click **OK**.
4. Click the **SQL Script** tab if you want to view the auto-generated SQL script.
   Notice that the parameter you added (id) is placed in the parameterized
query's WHERE clause.

**Including parameters in the FROM clause of parameterized queries:**

Follow these steps to include a parameter in the FROM clause of a parameterized
query.

**Procedure**
1. Right-click at an appropriate 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 opens in the right pane.

3. Drag the desired transformation that requires input parameters and drop it onto the **Model** panel.
   The **Input Parameters for** window opens.
   In this window, the upper section displays the input parameter(s). The icon next to a parameter represents the type of the parameter, such as INTEGER, STRING, and so on. The lower section of this window displays the definition and value corresponding to the parameter selected in the upper section. To display the parameter’s definition, click the **Show Definition** button.
   When you click the **Show Definition** button, the window displays the parameter’s name and type.
   In the **Value** group box, all the options are mutually exclusive: **Null** is for specifying a null value for the parameter. If it is disabled, it means that the parameter may not be null. **Literal** accepts a value for the parameter 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 is zero for a numeric, an empty string for a character, and the current date for a date.

4. Select the **Query Parameter** radio button, type the name ticker, and click **OK**.
   Do not include string values in quotes. The parameter’s name must be unique and begin with an alpha character, can contain any number of alpha-numeric characters, can include an underscore, and must not contain a space.

5. Click **OK** to save the entry and close the window.
6. If you want to view the auto-generated SQL script, click the **SQL Script** tab.

---

### Physical stored procedures

IBM Cognos Virtual View Manager supports the introspection of data sources that contain stored procedures, and these are referred to as physical stored procedures.

In this document, the terms physical stored procedure and stored procedure are used interchangeably. Currently, only the stored procedures written in SQL are supported.

A physical stored procedure may have scalar parameters and it may also return one or more cursors. A physical stored procedure returns a cursor to represent a complex output that is in tabular form.

Manual specification of cardinality is possible on stored procedures in physical data sources.

During the introspection of a stored procedure, the cursors are not automatically introspected and the direction of the scalar parameters may not always be clearly identified. Therefore it is sometimes necessary that the user has to manually specify the direction of the parameters and also define the cursor. Users are expected to know the number of cursors in a stored procedure and the structure of each cursor that they want to use.

When using a stored procedure in a Virtual View Manager view, these rules apply:
- A stored procedure that returns only one cursor can be used within a view.
A stored procedure that returns more than one cursor or a stored procedure that also has scalar parameters cannot be included in a view.

Virtual View Manager supports the introspection of stored procedures in the following database types: IBM DB2, Oracle, Microsoft SQL Server, and Sybase. Cursors are not introspected during the introspection of IBM DB2, Microsoft SQL Server, and Sybase, whereas in the Oracle data source the cursor type is unidentified and it is introspected as of the type OTHER. Without having to change the cursor's signature, you can still execute the Oracle stored procedure but the values are interpreted as binary data. But if you edit the stored procedure and define the cursor's signature, the output of the executed stored procedure will be displayed in a tabular form.

The rest of this section describes how to manually edit stored procedures and define cursor signatures.

About editing stored procedures

This section describes how to manually edit a stored procedure's signature.

Stored procedures in some data sources

During the introspection of some data sources, the direction of the scalar parameters in stored procedures maybe unidentified, and therefore you may have to manually specify the direction of those parameters.

This section describes how to specify the direction and define cursor signature. The stored procedure discussed here has two scalar parameters (one input and one output) and it returns a single cursor which has seven columns.

Specifying the parameter direction:

Follow these steps to specify the parameter direction.

Procedure

1. Open the desired stored procedure.
   You can double-click the stored procedure or right-click on it and select Open. The editor opens on the right.
   If the direction of an introspected parameter is unknown — whether it is IN, INOUT, or OUT — the icon next to the parameter will look as if it was broken.
2. Select the Design Mode check box in the editor's toolbar.
   This selection puts the editor in the design mode, so you can start editing the parameters.
3. Select the parameter whose direction is unknown, and use the direction arrows buttons on the toolbar for specifying the direction.

Manually defining cursors for projection:

Follow these steps to manually define a cursor for projection.

Procedure

1. Repeat the steps in the task "Specifying the parameter direction."
2. Click the Add button, and select Complex > CURSOR.
   A new parameter cursorParam of type CURSOR is added.
3. Change the direction of cursorParam to be OUT.
To do this, click the arrow next to cursorParam and use the direction-arrows buttons on the toolbar to change its direction to be OUT.

4. While cursorParam is selected, click Add, and select **Time > TIMESTAMP**. A new parameter is automatically placed under cursorParam.

5. Rename the newly added parameter to date, and click Enter.

6. Add the following parameters as children for cursorParam in the following order and specify each parameter's type as noted. Each parameter must be added individually. Once you specify its type, the parameter is automatically placed under the last child added.
   - 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 about the cursor's columns and their data types, you can use the Design By Example tool which helps you select the parameters, as described in the following task. Note that when you design a cursor using Design By Example all previously defined parameters for the cursor will be overwritten.

7. Select the parameter named RETURN_VALUE, and delete it.

   To delete RETURN_VALUE, you can use the main toolbar button (×), or right-click RETURN_VALUE and select Delete.

8. Save the stored procedure.

**Designing cursors by example:**

Follow these steps to design a cursor by example.

**Procedure**

1. Follow steps 1 to 5 in the task “Manually defining cursors for projection” on page 117.

2. Save the stored procedure after specifying the direction for the scalar parameter whose direction is unknown.

3. Click the Design By Example button on the stored procedure editor's toolbar.

   The Enter Values for Input Parameters window opens.

4. Supply a value for the scalar input parameter, and click **OK**.

   The Design By Example window opens.

   This window displays all the columns in the cursor.

5. Click **OK**.

   All the cursor columns are included for projection in the stored procedure.

   You can rename any of the parameters/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. Select the parameter named RETURN_VALUE, and delete it.

   To delete RETURN_VALUE, you can use the main toolbar button (×), or right-click RETURN_VALUE and select Delete.
7. Save the stored procedure.

**Editing stored procedures that return a single cursor**

The stored procedure discussed in this section is in the Northwind data source in Microsoft SQL Server database, and has 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 be found always in the same order (unless the stored procedure has changed since introspection.)

**Procedure**

1. Open the stored procedure by double-clicking the stored procedure or right-clicking it and selecting Open.
   The editor opens on the right. Notice that this stored procedure has one input parameter and one output parameter.
   The RETURN_VALUE parameter has to be deleted after you introspect the stored procedure.

2. You can design the cursor manually or by using the Design By Example tool, as described in “Stored procedures in some data sources” on page 117.

**Editing stored procedures that return two cursors**

This section describes how to edit a stored procedure that doesn't have any scalar parameters but returns two cursors.

**Procedure**

1. Open the stored procedure by double-clicking the stored procedure or right-clicking it and selecting Open.
   The editor opens on the right. Notice that there is no scalar parameter in this stored procedure.

2. Select the Design Mode box.
   This selection puts the editor in the design mode, so you can start adding and editing parameters.

3. Click the Design By Example button 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.

4. Click OK to include both the cursors for projection, and save the stored procedure.

**Rebinding procedures**

A procedure depends upon one or more underlying sources, and the procedure is considered bound to those underlying sources.

A procedure can be bound as follows:
- A SQL script can be bound to a table or a procedure.
- A transformation can be bound only to hierarchical data (XML or WSDL).
- A packaged query can be bound only to a data source.

Rebinding a procedure is useful:
- If anytime after creating a procedure with its sources you decide to rebind the procedure to different sources, you can do that.
- If there's an error in executing a procedure because a source with which the procedure was initially bound doesn't exist anymore and that you want to rebind the procedure with a new source.

**Procedure**

1. Select **Resource > Show Rebind Panel** to enable rebinding, and double-click the desired procedure.
   
   The **Rebind** panel opens in the lower section of the procedure editor, and displays the immediate dependencies as determined by the last successful saved version of the procedure. If a procedure has never been successfully saved, the **Rebind** panel will be blank accordingly.

2. In the **Rebind** panel, click the desired source displayed in the left pane.

3. In the **Rebind** window, specify the desired source with which to rebind the script, and click **OK**.

4. Save the procedure.

   If you make any modification to the procedure or its dependencies, the binding is automatically updated when you save the script and the Rebind panel reflects the change accordingly. For example, if your procedure has a data source dependency named `ds_orders`, and you rename `ds_orders` to `ds_orders_renamed` and save the script, the **Rebind** panel would list the dependency as `ds_orders_renamed`.

---

**About setting transaction options for procedures**

IBM Cognos Virtual View Manager can ensure consistency of procedure execution results by specification of a single execution per transaction given the same input parameters.

From the Info tab of the procedure definition the designer may check the box labeled Execute only once per transaction for each unique set of input values to ensure consistent results during a single execution.

When enabled, procedures executed within a transaction get executed only one time per unique set of input parameter values. If the procedure is executed more than one time with the same variant then the result set from the first execution is used. Different input values (variants) get retrieved normally. Results from these procedure variants are only valid for the lifetime of a transaction and discarded on transaction completion unless caching is enabled.

In addition to ensuring transactional integrity, repeated calls to the data source are minimized.

For example, suppose that a procedure has an input parameter as follows:

```plaintext
getAccountHistory(IN acctNum INTEGER, OUT results CURSOR)
```

If you call this procedure for the first time in a transaction with the integer 5 as the input value, the procedure will be executed and results will be returned. If you call the same procedure again with the same input value 5 in the same transaction, the procedure will not be re-executed, but the results of the previous call will be
returned. But if you call the same procedure with a different input value, say 6, the procedure will be re-executed and the results from the re-execution will be returned.

Note: When transaction procedure caching is enabled, procedure execution results are cached temporarily to memory (or to disk when the FORCE_DISK option is used) until the transaction is completed. If the procedure results are unusually large or the transaction remains unresolved for an inordinate amount of time, then this may impact performance.

About executing procedures

You can execute a procedure within IBM Cognos Virtual View Manager, or via a JDBC/ODBC client program or Web Service.

About executing procedures in Virtual View Manager

The basic task of executing a procedure is simple.

To execute a procedure, you open the procedure, click the Execute button (in the editor, and supply values for input parameters if the procedure has any.

This section gives examples for executing a parameterized query, an XSLT transformation, and physical stored procedures that return scalar parameters and cursors.

Executing parameterized queries

When you execute a query that has input parameters, you are prompted to supply values for the input parameters that have been added when the parameterized query was designed.

The parametrized query used here has two parameters—id and price—one (id) included in the SELECT clause and the other (price) in the WHERE clause.

Procedure

1. Double-click to open the parameterized query, and click the Execute button in the editor.
   
   The Enter Input Parameters window opens.
   
   The upper section displays the input parameter(s). The icon next to a parameter represents the type of the parameter, such as INTEGER, STRING, and so on. The lower section of this window displays the definition and value corresponding to the parameter selected in the upper section. To display the parameter’s definition, click the Show Definition button. When you click the Show Definition button, the window displays the parameter’s name and type.
   
   In the Value group box, the options are mutually exclusive: Null is for specifying a null value for the parameter. If it is disabled, it means that the parameter may not be null. Literal accepts a value for the parameter selected in the upper section.
   
   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 is zero for a numeric, an empty string for a character, and the current date for a date.
   
   2. Supply a value for the parameter by selecting the Null button to set the parameter value to be null, or supplying a value in the Literal field.
For this example, select a parameter, select the **Literal** button, supply a value. Let's type the values 5 for id and 50 for price. When the parametrized query is executed, names of all the products with id 5 and price less than $50.00 will be displayed.

3. Once you finish supplying values for both the parameters, click **OK**.
   If the executing task is successful, the Result For result panel displays the result set, fifty rows at a time.

4. In the Result For result panel:
   - To view the next fifty rows in a large data set, click the **Next** button.
   - To view the details of a specific row, select the row, and click the **Details** button.
   - To save the results, use the **Save to File** button.
   - To clear the display in the tab, click the **Clear** button.

### Executing transformations

Follow these steps to execute a transformation.

**Procedure**

1. Double-click the desired transformation to open it, and click the **Execute** button on the toolbar.
   If there is no need for an input parameter value, the procedure is executed.
   If there is an input parameter for which you need to supply a value, the **Enter Values for Input Parameters** window displays prompting you for an input value.
   When you click **Show Definition**, the definition of the parameter is displayed.
   In the **Definition** group box:
      - The entry in the **Name** field is the name of the XML element.
      - The entry in the **Qualified Name** field is the URN of the element.
      - The entry in the **Type** field refers to the type of the element.
      - The next two fields **Min Occurs** and **Max Occurs** refer to the allowed minimum and maximum number of occurrences respectively of the element.
      - If **Min Occurs** is 0 (zero), there are no instances of the element, and that is indicated by the element name being gray. You may add instances of this element up to the number of times defined by the corresponding **Max Occurs** value. To add an occurrence of this element, right-click the element name in the upper panel, and click **Add <element name>**. If **Max Occurs** is unbounded, you can add this element any number of times. To add an occurrence of this element, right-click the element name in the upper section, and click **Add <element name>**.

2. To supply value for an input parameter, select the parameter, select **Null** button to set the element's value to be null, or supply a value for the element in the **Literal** field.
   Do not include string values in quotes.

3. Click **OK** to save the entry and close the window.

4. If the executing task is successful, the Result tab displays the result set, fifty rows at a time.
   In the Result panel:
      - To view the next fifty rows, click the **Next** button.
      - To view the details of a specific row, select the row, and click the **Details** button.
To save the results, use the Save to File button.
To clear the display in the tab, click the Clear button.

Note: If the executed transformation is an XQuery transformation, you can save the result (using Edit File) as an XML file and view it in a browser.

About executing stored procedures
This section describes how to execute a stored procedure that returns one or more cursors.

There can be any number of cursors in a stored procedure. The stored procedure used here doesn't have any scalar parameters, and it returns two cursors both of which are manually defined via the Design By Example window.

Executing stored procedures that return a single cursor:
Follow these steps to execute a procedure that returns a single cursor.

Procedure
1. Edit the desired stored procedure.
2. Save the stored procedure, and click the Execute button on the toolbar.
   Because the stored procedure has a scalar input parameter, the Enter Values for Input Parameters window opens.
3. Supply a value for the scalar parameter, and click OK.
   The stored procedure is executed and the cursor output is displayed.

Executing stored procedures that return two cursors:
Follow these steps to execute a stored procedure that returns two cursors.

Procedure
1. Edit the desired stored procedure.
2. Save the stored procedure, and click the Execute button on the toolbar.
3. Click the Display button next to the cursor whose output you want to view.
4. To return to the main display panel from the output display, click the Result tab.
5. Click the Close button next to the Display button that has been disabled.
   You can view the results of any cursor in any order. However, due to JDBC driver constraints, all the rows of the first-opened cursor needs to be buffered before the rows of the next cursor are returned. Therefore, if you are interested in the results of the second cursor, you can close the first cursor to disable buffering to enhance performance.

Executing procedures via JDBC
This section contains a sample Java code to execute a stored procedure that returns two cursors.

The name of the stored procedure used in the sample code is SP_D1. The username and password used to log into the server are guest and password respectively.

```java
import java.sql.*;
public class MultipleCursors {
```
private static final String VVM_URL = "jdbc:cognos:dbapi@localhost:9401?domain=cognos&dataSource=ds";
private static final String VVM_DRIVER = "cs.jdbc.driver.CompositeDriver";
private static final String USER_NAME = "guest";
private static final String PASSWORD = "password";
public static void main(String[] args) {
    try {
        Class.forName(VVM_DRIVER_NAME);
    } catch (ClassNotFoundException ex) {
        ex.printStackTrace();
        return;
    }
    try {
        executeProcedure();
    } catch (SQLException ex) {
        ex.printStackTrace();
        return;
    }
    private static void executeProcedure() throws SQLException {
        Connection conn = DriverManager.getConnection(VVM_URL, USER_NAME, 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));
            }
        }
    }
}

Procedure
1. Publish the stored procedure as a Virtual View Manager database to the following directory:
   Virtual View Manage Data Services/Databases/ds
2. Write and compile the Java code, and save the class file in a directory.
3. From the directory where you saved the class file, run the following command:
   java -classpath <path_to_csjdbc.jar_file>/csjdbc.jar . MultipleCursors
   where MultipleCursors is the name of the class file, and csjdbc.jar is the name of the JAR containing the class file.
Executing procedures via a web service

Follow these steps to execute a procedure via a web service.

**Procedure**

1. Make sure that the procedure is available as a Web service operation on the IBM Cognos Virtual View Manager server, as in the following location, for example:
   
   Virtual View Manager Data Services/Web Services/<service_name>/
   <service_folder>/<port_folder>/<operation>
   
   If the procedure is not yet available as a Web service, publish it as a Web service.

2. Open a Web browser, and type the following URL:
   
   http://<host_machine_name>:9400/services/<service_name>.html

3. In the Connect to <host_machine_name> window, enter a user name and password to connect to the server, and click OK.

   A list of available services is displayed.

4. Click the desired operation.

5. Supply the user name and password again, and click OK.

6. Supply values for the input parameters, if any, and click the **Execute Operation** button.

   The procedure/operation is executed and the service request and response are displayed in the browser.
Chapter 6. Definition sets

This chapter describes definition sets and shows how to use them. This chapter assumes that you are familiar with the information presented in the preceding chapters.

About definition sets

A definition set is a set of definitions you create. 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/views using the definition set. You cannot publish a definition set. The definition set you create is re-usable and can be shared with any user with appropriate access privileges in the system.

IBM Cognos Virtual View Manager supports two types of definition sets—XML-based definition set and SQL-based definition set—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

Virtual View Manager has an editor for building and editing definition sets. The definition set editor opens automatically when you create a definition set. See
Steps to create a definition set for details. At any time later, you can open the definition set editor by double-clicking the desired definition set in the resource tree.

The XML definition set editor

The XML definition set editor lets you create your own XML schema or generate the schema from an existing XML instance file.

This editor has three panels: XML Schema, XML Schema Text, and Info which are described in the following sections.

XML schema panel

This panel displays the XML schema, element declarations, and type definitions from the XML schema text.

The XML schema text is provided in the XML Schema Text panel. The Show Details button may be toggled to show or hide the properties panel. Any schema object may be selected to view the properties of that object. Use the Show All Properties check box to display all of the properties of the element selected in the XML schema. When not selected, only the basic properties of the selected element are displayed.

XML schema text panel

In this panel, you can type your schema or insert the schema from a file, or upload an XML instance file from which to generate the schema.

Info panel

This panel displays the details about a definition set.

The details are user-provided name (Name) which is fully qualified, resource type (Type), owner name (Owner), the domain to which the owner of the definition set belongs (Owner Domain), and annotation on the resource (Annotation). Information in this panel can be edited only by a user with the WRITE privilege on the definition set.

The Info panel in an XML definition set editor also requires input in the following fields: Base URL for Schema Locations, External Schema Location, External No Namespace Schema Location. Information supplied in these fields is used to resolve the schema locations specified in a schema’s import declarations.

Import declarations in a schema may contain a namespace or a schema location. If they contain a schema location it may be relative or fully qualified, as in the following examples:

```xml
<xs:import namespace="http://fileName1">
<xs:import schemaLocation="fileName1.xsd">
<xs:import schemaLocation="file:///myschemas/fileName1.xsd">
```

Base URL for schema locations

The entry in this field is used to resolve relative schema locations in import declarations.
If the schema location in an import declaration is a relative path name, then it will be resolved with respect to the **Base URL for External Schema Locations**. For example, if the schema name is `fileName1.xsd` and the **Base URL for Schema Locations** is `file:///myschemas`, then the server will look for a resource in: `file:///myschemas/fileName1.xsd`.

**External schema location**

The entry in this field is used to resolve imported schemas that are referred to by namespace, not by location.

It may contain one or more pairs of XML Namespace URIs followed by a fully qualified schema location URL. For example, suppose your schema contains the following import declarations:

```
<xs:import namespace="http://fileName1"
<xs:import namespace="http://fileName2"
```

Then you might have an External Schema Location entry that looks as follows:

```
http://fileName1 file:///myschemas/fileName1.xsd
http://fileName2 file:///myschemas/fileName2.xsd
```

**External no namespace schema location**

Some XML schema definitions have no target namespace. If they import another schema that has no target namespace, they must set this field to the name of a schema file that contains the imported schema definition.

---

### The SQL definition set editor

The XML definition set editor lets you create your own SQL type definitions, exceptions, and constants.

This editor has four panels: **Types**, **Exceptions**, **Constants**, and **Info**.

- The **Types** panel lets you define SQL data type definitions.
  - The **Add** button is enabled when you create a SQL definition set. Use the **Add** button to add types, exceptions, and constants when you design the definition set.
- The **Exceptions** panel lets you define exceptions.
- The **Constants** panel is available for defining constants.
- The **Info** panel displays the details about a definition set: user-provided name (**Name**) which is fully qualified, resource type (**Type**), owner name (**Owner**), the domain to which the owner of the definition set belongs (**Owner Domain**), and annotation on the resource (**Annotation**). Information in this panel can be edited only by a user with the WRITE privilege on the definition set.

---

### The definition set editor toolbar

The following table lists the definition set editor's toolbar buttons (in alphabetical order) and their functionality:

<table>
<thead>
<tr>
<th>Button</th>
<th>Label</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add" /></td>
<td>Add</td>
<td>Adds a new Type, Exception, or Constant to an SQL-type definition set</td>
</tr>
</tbody>
</table>
About creating definition sets

You can create a definition set anywhere except within IBM Cognos Virtual View Manager Data Services.

For other details, see "Where can you create a resource?" on page 11.

This section describes how to create definition sets of SQL type and XML type. Note that an XML definition set can be created either from a schema or from a raw XML file instance.

Creating definition sets

Follow these steps to create a definition set.

Procedure

1. Right-click at an appropriate location in the resource tree, and select New Definition Set.
   Alternatively, at the chosen location you can select File > New > Definition Set.
2. In the input dialog, type a name for the definition set.
3. Specify a type for the definition set—SQL or XML—in the Type drop-down list.
4. Click OK.

Results

The definition set is added to the specified location in the resource tree, and the editor for the definition set opens in the right pane.

Now you're ready to add definitions to the set.
Creating SQL-type definition sets

Follow these steps to create an SQL-type definition set.

**Procedure**

1. Follow steps 1 - 2 described in the task "Creating definition sets" on page 130.
2. To add type definitions, click the **Types** tab in the editor, and click **Add**. In the menu list that opens, specify the desired type or use the **Browse** button to upload an existing definition set. You can include SQL-type definitions as well as XML-type definitions in an SQL-type definition set. The following steps describe how to create your type definitions from scratch.
   - Click **Add**, and select the desired data type.
     
     The type is added as a **NewTypeDefinition** entry in the **Name** column.
     
     You can rename the newly added definition by typing the new name when the definition is created. Or, any time later, you can right-click the definition name, select **Rename**, and type the new name.
     
     If the type is complex, you can move a definition up/down/in/out) by using the navigation arrows in the editor's toolbar.
   - Rename **NewTypeDefinition** according to your choice.
   - Repeat steps 2.1 and 2.2. In step 2.2, rename the newly added type as lastname.
   - Click **Add** and select **String > VARCHAR**, and rename the type as **SSN**.
   - Save the definition set.

Now, the definition set editor should display the newly added entries as follows:

![Figure 2. The definition set editor window displaying newly added entries](image)

The following steps describe how to upload type definitions from another definition set.

Note that you can include SQL-type definitions as well as XML-type definitions in a SQL-type definition set.

- In the definition set editor, click **Add**, and select **Browse**.
  
  The **Add Definition Type** window opens.
Locate and select the desired definition set in the left panel. Select Type in the Show drop-down list in the right panel. Click OK.

The new type definition is added to the Name column in the Types panel. Only the type is uploaded as is, but you need to rename the type definition.

3. To add exceptions, click the Exceptions tab and add exceptions as you added types.

4. To add constants, click the Constants tab. Add constants as you added types and exceptions.

5. To supply a value for a constant:
   - Select the constant.
   - Right-click below Value, and select Change Value.
   - In the input window, supply a value for the constant, and click OK.

5. (optional) Click the Info tab and supply an annotation for the definition set.

6. Save the definition set.

Results

You can refer to the types, exceptions, and constants in a definition set from any procedure and also share the definition set with any user that has appropriate privileges in the system.

See also “Using SQL-type definition sets” on page 135.

Creating XML-type definition sets from XML schemas

Follow these steps to create an XML-type definition set from an XML schema.
Procedure

1. Follow steps 1 - 2 described in the task “Creating definition sets” on page 130.

2. In the editor that opens on the right, select the XML Schema Text tab for defining your XML schema.
   
   You can either type your XML schema in this panel or upload an existing XML schema by using the Insert From File button in the editor's toolbar.

   This example uses a file named addressSchema.xml.

   ![XML Schema File Example](image)

   Figure 4. Example of an XML Schema file

3. Once you have the desired XML schema in the XML Schema Text panel, save it.
   
   You can edit the schema as necessary.

4. Use the Parse XML Schema ( readiness ) in the toolbar to parse the schema, after editing the schema.

   Parsing the schema will parse the schema into Element Declarations and Type Definitions, which you can find in the XML Schema panel.

   Use the Show Details button to view properties of selected XML nodes.
5. Click the **Info** tab to open the **Info** panel.

6. The **Base URL** field is optional if the absolute path of the schema definition set will be set in the **External Schema Location** field. Use the **Base URL for Schema Locations** field, to specify the base URL to the XML schema. File, HTTP/HTTPS, and FTP protocols are supported. It can be a local path (example: file://C:/MySchemaDefinition.xsd) or an HTTP/HTTPS path (example: http://www.w3.org/XML/1998/namespace).

   file:///myschemas/fileName1.xsd

   The **Base URL** field is automatically filled if an existing schema was uploaded.

7. In the **External Schema Location** field, specify the path to the XSD file if you want to use it to resolve the schema.

   You can let the system resolve the schema, if no external XSD files are required to resolve the schema.

   The syntax for the location is as follows:

   Set the absolute path of the external schema location.

   Example: file:\\c:\vvm\bugs\Schema_With_Namespace\xml-ns.xsd

   `<namespace> <location> [<namespace> <location>]`  

   Example: http://fileName1 file:///myschemas/fileName1.xsd

8. The **External No Namespace Schema Location** is an optional field for use when the XML schema definitions does not have a target namespace. If another schema utilizes a file that has no target namespace, enter the path and name of the schema file that contains the imported schema definition in the **External No Namespace Schema Location**.

9. Save the definition set.

**Creating XML-type definition sets from XML file instances**

   Follow these steps to create an XML-type definition set from an XML file instance.
Procedure
1. Follow steps #1-2 described in the task “Creating definition sets” on page 130.
2. In the editor that opens on the right, select the XML Schema Text tab to open the XML Schema panel.
3. Click the Create Schema From XML Instance toolbar button ( ).

4. In the Select the XML instance file field, supply the full path to the XML file instance.
   You can also use the Browse button to locate the XML file.
   This should be an actual XML file, not a schema.
5. In the Select the XML Schema output directory, supply the full path to a directory to store the output schema.
6. Press OK.
7. In the editor, view the schema in the Elements panel.
8. Save the definition set.

About using definition sets
This section gives two examples showing how to use an SQL-type definition set and an XML-type definition set.

Using SQL-type definition sets
The procedure described here refers to the data types defined an SQL definition set.

The SQL definition set is named mySQLDef:

<table>
<thead>
<tr>
<th>Name</th>
<th>Base Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstname</td>
<td>CHAR(255)</td>
</tr>
<tr>
<td>lastname</td>
<td>CHAR(255)</td>
</tr>
</tbody>
</table>

Procedure
1. Create an SQL script, named callSQLDef.
   Select File > New > SQL Script.
2. Type the following script in the SQL Script panel
PROCEDURE callSQLDef(OUT Name VARCHAR(512))
BEGIN
DECLARE FirstName /users/cognos/admin/mySLDef.firstname;
DECLARE LastName /users/cognos/admin/mySLDef.lastname;
SET FirstName = 'myFirstName';
SET LastName = 'myLastName';
SET Name = CONCAT(CONCAT(TRIM(FirstName), ' '), TRIM(LastName));
CALL Print(Name);--prints to the console
END

3. Execute the script to see the result (next screen).

About using XML-type definition sets
The procedure used in this section uses an XML definition set.

The XML definition set is named XMLLib_Composite, and has the following XML schema and elements (rendered in bold type) within the schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" targetNamespace="http://www.cognos.com/services/webservices/system/admin/resource"
xmlns:ns="http://www.cognos.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="Address" type="ns:Address"/>
</xs:schema>
```
Using XML-type definition sets

The procedure described here supplies values to the elements in the XML schema in the definition set XMLLib_Composite.

For more information, see "Creating definition sets" on page 130.

Procedure

1. Create a SQL script using Select File > New > SQL Script at an appropriate location in the resource tree.
2. Type the following script in the SQL Script panel of the script editor:

   ```sql
   PROCEDURE ScriptComposite(OPT CompanyAddress /shared/sources/definitionSets/XMLLib_Composite."{http://www.cognos.com/services/webservices/system/admin/resource}Address")
   BEGIN
     CompanyAddress = '<ns:Address xmlns:ns="http://www.cognos.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:Address>';
   END;
   
3. Execute the script to see the result.
4. Click the Details button to view the XML display.
Chapter 7. Triggers

This chapter describes the functionality of triggers. This chapter assumes that you are familiar with the information presented in the preceding chapters.

These topics are covered in this chapter:
- "About triggers"
- "The trigger editor"
- "Creating triggers" on page 141

About triggers

A trigger is a resource that initiates a specific system activity to be performed when specified conditions occur without requiring user interaction.

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 in response to a system event, such as a request failure or a data source being up or down, can be triggered.
- Procedures can be executed without sending notifications.
- Views and procedures can be executed and the results can be sent to one or more recipients.
- Custom tasks on a data source, such as re-introspection or statistics gathering, can be triggered instead of generating only e-mail in a fixed format from a scheduled data source that is re-introspected or scanned for statistics data gathering.

The trigger editor

If you want to create a new trigger for a resource, you use the trigger editor in IBM Cognos Virtual View Manager. The trigger editor opens automatically when you create a trigger as you would create any other resource. At any time later, you can open a trigger’s editor by double-clicking the trigger’s name in the resource tree.

The following sections describe the panels in the trigger editor.

Note: Note that if scheduled resources from a previous release are imported into the current release, those schedules are displayed as triggers in the resource tree.

The trigger editor has two panels—the Condition panel and the Info panel—which are described in this section.

Condition panel

The Condition panel in the trigger editor has the Enable Trigger check box which must be selected in order to activate a trigger.
In addition, it has four panes in which you specify the trigger information:

- **Condition Type** pane (upper-left)
- **Condition** pane (upper-right)
- **Action Type** pane (lower-left)
- **Action** pane (lower-right)

The panes on the right display the details corresponding to the item selected in the left pane.

**Condition types for a trigger**

Timer event, system event, and user-defined event are the currently supported trigger condition types.

- Timer event refers to the standard scheduling timer, and lets you specify the start time and frequency to schedule the desired action. 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 e-mail with the results of a procedure/view execution.

- System event is any system-generated event. User tasks can listen for such events. Two built-in procedures—GetEnvironment and SetEnvironment—are available in the `/lib/util/` directory, which 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 the current events for you to select one. Current release supports only a selected number of system events. Note that making a trigger based on a system event will not enable a disabled event. For example, you cannot enable a disabled data source through a trigger.

- User-defined event is any user-defined event. A built-in procedure named GenerateEvent is available in the `/lib/util/` directory, which 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 desired action asynchronously.

**Action types for a trigger**

Executing a procedure, gathering statistics on a data source, re-introspecting a data source, and sending e-mail are the trigger action types that can be triggered under the conditions described in the preceding section.

- The Execute Procedure action lets you specify a procedure to be executed without sending any notification. For further details, see the next section.

- The Gather Statistics action lets you specify a data source whose data statistics are to be gathered. For further details, see the next section.

- The Reintrospect Data Source action lets you specify a data source to be re-introspected. For further details, see the next section.

- The Send E-mail action lets you specify a procedure or view to be executed, and send the results through e-mail to one or more specified recipients. For further details, see the next section.

**Condition pane**

The contents of the Condition pane depend on the condition type selected.

The condition types are as follows:
Timer event conditions
You specify the frequency for when the trigger occurs. See “Creating timer event triggers” for more details.

System event conditions
You specify the system event name. See “System event triggers” on page 144 for more details.

User-defined event conditions
You specify the user-defined event name. See “User-defined event triggers” on page 143 for more details.

Action pane
The contents of the Action pane depend on the action type.

See “Action types for a trigger” on page 140 and “Creating timer event triggers” for details about what information is specified.

Info panel
The Info panel displays the following details about the current trigger:
• Name—user-provided name and location.
• Type—resource type.
• Owner—owner name.
• Owner Domain—domain to which the owner of the trigger belongs.
• Maximum Number in Queue—displays the number of times a trigger can be queued simultaneously before duplicates are dropped. The default value is 1 (one), and no negative number is permitted. The default value 1 means that the queue for this trigger cannot contain more than one extra event other than the one currently being processed. With a value of 1, if the trigger repeats to start before the previous one finishes its task, the additional starting is ignored. With a value of 2 (two), you can have one trigger running and when that one finishes a second will start immediately. If you set the maximum number in queue to be 6, for example, and have 6 refresh failures at once, you will receive 6 e-mail notifications. Larger number of queues for a trigger would allow less event loss under heavy load at the potential cost of memory.
• Annotation—annotation on the trigger.

Creating triggers
You can create a trigger just as you would create another IBM Cognos Virtual View Manager resource. However, note that a trigger cannot be published.

This section describes how to create three kinds of triggers:
• Timer event triggers
• User-defined event triggers
• System event triggers

Creating timer event triggers
This section describes how to create a timer event trigger. Specification of information for the various Action panes is also described.
Procedure

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.
   The trigger editor opens on the right. See the screenshot in the section "The trigger editor" on page 139. Notice the four sections—Condition Type pane (upper-left), Condition pane (upper-right), Action Type pane (lower-left), and Action pane (lower-right)—and their contents.
3. Enable the trigger by selecting the Enable Trigger check box.
4. Select Timer Event, if it’s not already selected, in the Condition Type drop-down list.
5. Specify the schedule in the Condition pane on the right:
   - To schedule the event to be executed once, select Execute Once.
   - To schedule the event periodically, specify the frequency—every # minutes, hourly, daily, or weekly—by selecting the corresponding radio button.
   - To specify the time and date for view execution (once or periodically) use the corresponding Time/Date drop-down boxes.
     The date entered indicates the time at which the first occurrence of the scheduling event will occur. For example, if a daily event is set for 11:55 am three days in the future, it will run at 11:55 am in three days and then every day thereafter at 11:55 am
6. In the Action Type drop-down list, select the desired action to be triggered.
   The options are as follows, and the details are displayed in the Action section:
   - Execute Procedure—Lets you execute a procedure without sending any e-mail notification to anyone. If a procedure has input parameter(s), use the Edit button to open a window and supply the values for the parameter(s).
     To see how it works, select the built-in procedure: /lib/resource/TestAllDataSourceConnections
     This procedure has no input parameters.
   - Gather Statistics—Lets you specify a data source whose statistics data are to be gathered. Optionally, you can also specify one or more e-mail recipients to receive notification with the statistics data gathered.
     To see how it works, select the data source:/shared/examples/ds_orders
   - Reintrospect Data Source—Lets you specify a data source to be re-introspected. Optionally, you can also specify one or more e-mail recipients to receive notification with the result of the re-introspection. Select Report Only if you want only a report about the changes detected in the data source, and not save the changes. You can also choose not to receive any notification if no changes are detected in the data source by selecting Do Not Send If No Changes Detected.
     To see how it works, select the data source: /shared/examples/ds_orders
   - Send E-mail—Lets you specify a procedure or view to be executed, and receive notification with the output of the procedure/view execution. You can choose to have a summary of the notification by selecting the Include Summary check box. You can also choose not to receive any notification if there are no results by selecting Do Not Send If No Results.
     Before scheduling a view to execute, you should set the e-mail configuration via the Configuration window in Virtual View Manager. For details on setting the e-mail configuration, see the Configuration chapter in the IBM Cognos Virtual View Manager Administration Guide.
To see how the Send E-mail action works, select the transformation:
/shared/examples/productCatalog_Transformation

After creating the trigger, view the Triggers console in the Manager to check if the trigger was fired as scheduled. Also verify if the notification was sent to the mail recipient(s).

7. In the Action section, type and confirm the e-mail address(es) of the recipient(s) to receive notification, as needed.
   You can specify a list of e-mail addresses in the address fields by separating each one with a comma (,) or semicolon (;).

8. Click the Info tab, and in the Maximum Number In Queue field specify the maximum number of times this trigger is to be started. 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.

User-defined event triggers

First you 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.

Creating user-defined events

Follow this step to create a user-defined event.

Procedure

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 will create a custom event with the name runAReport. Whenever this procedure is executed, GenerateEvent will insert the custom event named runAReport into the trigger system.

Creating user-defined event triggers

Follow these steps to create a user-defined event trigger.

Procedure

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 names. For this example, supply the name as runAReport.
6. Continue with step 6-9 in the section "Creating timer event triggers" on page 141.
Testing user-defined event triggers
Follow this step to test a user-defined event trigger.

Procedure

Run the procedure CallsGenerateEvent that was created as described in "Creating user-defined event triggers" on page 143.

System event triggers

The following system events are supported as conditions for a trigger action: CacheRefreshFailure, CacheRefreshSuccess, and RequestFailure.

The built-in procedures—GetEnvironment and SetEnvironment—that are available in /lib/util/ directory 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.

Creating sample procedures to call GetEnvironment
Follow this step to create a sample procedure to call GetEnvironment.

Procedure

Create a sample SQL script procedure, named CallsGetEnvironment as follows:

```sql
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.

Creating system event triggers

Follow these steps to create a system event trigger.
**Procedure**

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 the desired system event in the **System Event Name** drop-down list on the right.
   You can also use a regular expression, for example, .*Failure, to match more than one similar system-event names.
6. In the **Action Type** drop-down list, select the desired action to be triggered.
   The options are described in step 6 in the section "Creating timer event triggers" on page 141.
   - **Execute Procedure**—To see how it works, specify the path to the procedure CallsGetEnvironment in the Action section.
   - **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. Continue with step 7 - 9 in the section "Creating timer event triggers" on page 141.

**Testing system event triggers**

Follow these steps to test a system event trigger.

**Procedure**

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. Manually refresh the cache of a broken view.
   The specified mail recipient(s) will receive e-mail notification and the result of running the path to procedure CallsGetEnvironment will be written to the log.
Chapter 8. Resource publishing

Resources are defined in a user workspace or in a shared workspace, but they will not be available for end-user view and consumption until they are published as either data services or web services with appropriate privilege settings. This chapter describes publishing of resources as a data services, and leaves the privileges that grant permission to view the resource.

These topics are covered in this chapter:
- "About resource publishing"
- "Types of published resources"
- "Locations of published resources"
- "Publishing resources" on page 149
- "Operations for published resources" on page 151
- "Access privileges for published resources" on page 151
- "Viewing the WSDL of web services" on page 151
- "Testing web services" on page 152

About resource publishing

Make data services available to client applications by publishing IBM Cognos Virtual View Manager Data Services. Resources may be published as either databases that may be queried via JDBC or ODBC, or they may be published as web services that may be called via HTTP, HTTPS, SOAP over Apache WSIF JMS, or SOAP over TIBCO JMS with the appropriate protocols.

Various protocols may be used to publish Virtual View Manager data services. Resources may be consumed via JDBC, ODBC, or SOAP.

Types of published resources

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 procedures (SQL scripts, Java procedures, transformations, parameterized queries, and packaged queries). You can also publish an entire data source. Note that the structure of the data source is preserved in the published version.

For the sake of reference in this document, let us call the publishable sources original resources. When you publish an original resource, the newly created resource is called a published resource. Each published resource refers to an original resource that resides elsewhere in the system.

Note: Any time an original resource is modified, the corresponding published resource reflects the change. Also, you can access and use an original resource through the corresponding published resource.

Locations of published resources

Resources are published to the Virtual View Manager Data Services node in the resource tree.
As is discussed in the Getting Started chapter, the resource tree has four primary nodes **Virtual View Manager Data Services**, **My Home**, **Shared**, and **localhost**. The nodes **My Home**, **Shared**, and **localhost** are development workspaces that may be organized according to project needs or user convenience. Conversely, the **Virtual View Manager Data Services** node is a published interface that should be organized for the convenience of the JDBC/ODBC client introspecting on published data tables. Likewise web services are organized as WSDL services and ports for the convenience of the web services client.

Resources created in **My Home** (a shortcut to the current user’s workspace), **Shared** (also a shortcut), or **localhost** may be published in the IBM Cognos Virtual View Manager Data Services node as either a database or a web service.

- Directly to a database or to a schema in a database. You can publish a data source only to a database. See “Databases” for more information.
  
  Client applications can access these resources via JDBC/ODBC.
- To the operations folder in a web service. See “Web services” for more information on publishing to this area.
  
  Client applications can access these resources through a web service (SOAP).

**Note:** A procedure published in a Virtual View Manager database is a stored procedure, whereas a view published in a web service is a web service call without arguments.

For details on how to publish, see “Publishing resources” on page 149.

### Databases

Databases is a container within Virtual View Manager Data Services. You create Virtual View Manager databases, catalogs, and schemas in this container to publish tabular data and procedures. You can create as many databases and schemas as you need to publish your resources.

You can query the published tabular data as you would query any normal database.

For details on how client applications can query published resources, see Chapter 12, “Client interfaces,” on page 223.

### Web services

Web Services is a container directory within Virtual View Manager Data Services. You create web services (and other folders) within this container. Within a web service, publish tables and procedures as links to the actual resources. These published links are placed in the operations folder within the service. And finally create ports to expose the WSDL interface that is automatically generated for all of the published links in the operations folder.

For details on how client applications can query and consume published resources, see Chapter 12, “Client interfaces,” on page 223.

### Publishing Virtual View Manager Data Services over a JMS connector or via SOAP over HTTP

The following steps show you how to design Virtual View Manager Data Services from the bottom up.
Before you begin

Prior to publishing resources using a SOAP/JMS binding, the connector between the Virtual View Manager server and the JMS broker must be configured. Creating and configuring the JMS connector above requires administrative rights and working knowledge of the current JMS via JNDI connector as implemented in your particular environment. Configuration of the JMS Connector is described in the IBM Cognos Virtual View Manager Administration Guide.

Procedure

1. Create and configure the JMS connector if one is to be used. For more information, see the IBM Cognos Virtual View Manager Administration Guide.

2. Publish tables, views, or procedures to the operations folder in the web service directory. The WSDL interface is derived from the links published to the operations folder.

3. Configure the service by creating and configuring port(s) for publishing the service interface using a SOAP/JMS binding and a JMS connector or by SOAP over HTTP/HTTPS. An HTTP port does not require a connector.

Publishing resources

The following steps show you how to publish a resource.

Procedure

1. Right-click the desired table, view, or procedure from the resource tree, and select Publish, or select the resource and use the Resource menu > Publish option.

   Multiple resources may be selected for bulk publishing by holding the Ctrl key and selecting multiple resources. Adjacent resources may be selected by holding the Shift key and making a selection.

   The Publish <ResourceName> window is displayed. Use it to create new folders, web services, IBM Cognos Virtual View Manager Web Services, operations folders, and ports.

2. Open the Web Services node.

   If a Virtual View Manager Data Service is not yet available (or a new one is desired), it must be created.

3. Enter a name for the new data service, and click OK.

   A WSDL definitions place holder is created within the new data service. Each WSDL interface generates a single class available over one or several ports.

4. Create (or select) a named service within from the Virtual View Manager Data Service node to host the published operations that will be hosted by your server.

   An operations folder will be created as a container to hold links to published resources.

5. Under Virtual View Manager Data Services, right-click Service, and click New Port.

6. Open the port to configure it for use.

   The Operations field shows the group of services to which the port will be bound. All resources published as links to a particular operations folder are made available to all of the ports bound to that service node. So if a service node has three different ports then the WSDL will make them available to all.
Deleting a published operation link has no effect on the original resource definition. Deletion of the operations link will remove the service from availability.

7. Select one of the three **Binding Profile Types**.

SOAP over HTTP, SOAP over WSIF JMS, and SOAP over TIBCO JMS are supported. When connecting a Virtual View Manager Data Service to another Virtual View Manager Data Service then any of the binding profile types can be selected. If the service is published over the Apache WSIF profile then the client must use the WSIF specification. Likewise if the service is exposed using TIBCO JMS the client will require implementation of TIBCO specifications.

The **Binding Profile Type** selection determines what configuration parameters are exposed for further specification.

For **SOAP over HTTP**, SSL/TLS may be toggled on and off. Authentication may utilize either Basic, WS-Security Username Token, or both authentication types together. Client may be required to use all selected methods of authentication or they may be allowed to use any of the methods allowed.

Pipeline message transformations are configured on the level of the operations folder. Refer to the Security chapter section on “Transport layer security (TLS)” on page 168 or “Message level security” on page 169 for more information.

For **SOAP over WSIF JMS** and **SOAP over TIBCO JMS**, only Auto correlation is currently supported. Other correlation types will be supported in future releases.

If the **Connector** field is blank, then you must configure the connector between the server and the JMS broker before this port is functional. See IBM Cognos Virtual View Manager Administration Guide.

In the SOAP/JMS specifications, SOAP messages are packaged in a JMS message with additional headers used for dispatching and interpreting the body of the message. These headers are defined in addition to the standard JMS headers needed for pure JMS applications. The standard JMS headers are also used, as long as their semantics are consistent with web services concepts.

Currently for SOAP over JMS only **Queues** are supported as Destination Types. The Queue destination is created in the JMS Broker namespace.

The port configuration of the SOAP over JMS binding type may specify authentication as either Basic or WS-Security with a Username Token or both authentication types together. The client may be required to use all selected methods of authentication, use any of the methods allowed just like the SOAP over HTTP, or use none of the authentication methods. Unchecked authentication check boxes enables anonymous usage of the services.

By contrast, JMS does not support SSL/TLS.

JMS message properties may be set for individual operations. Reply messages sent by Virtual View Manager may use message type and expiry. To set these properties, show the Advanced Properties using the button provided and add an operation.

The message type may be set to either **Bytes** which may be faster, or the default **Text** may be specified. At this time only Auto Acknowledgement is available, but client side acknowledgement will be available in the future. The expiry period for the message in the queue may be set, or it will default to 10000 milliseconds (10 seconds).
Operations for published resources

You can perform three operations on a published resource.

These three operations are the following:
• Access its underlying source.
• View the resource contents if it is a table. Execute the resource if it is a procedure.
• Rebind to a difference source. See “Rebinding a view” on page 89 for details.

Accessing the underlying source of published resources

Follow these steps to access the underlying source of a published resource.

Procedure
1. Double-click the desired published resource to open it.
2. In the editor that opens on the right, click the Info tab to open the Info panel.
3. Click Open Source.

Viewing the contents of published views

Follow these steps to view the contents of a published view.

Procedure
1. Double-click the desired published view to open it.
2. Click the Show Contents toolbar button.
3. Alternatively, you can right-click at the view’s name in the resource tree and select Show Contents.

Executing published procedures

Follow these steps to execute a published procedure.

Procedure
1. Double-click the desired published procedure to open it.

   Note: Note that a web service operation is also considered a procedure.

2. In the editor that opens on the right, click the Execute toolbar button.

Access privileges for published resources

You need the right set of access privileges on a published resource and its original resource in order to use the published resource. A published resource has its own READ and WRITE privileges to let you manipulate it in IBM Cognos Virtual View Manager, but it does not have its own run-time privileges (EXECUTE, SELECT, INSERT, UPDATE, DELETE).

Viewing the WSDL of web services

IBM Cognos Virtual View Manager provides an option to view the WSDL underlying a published web service and see how it would display to a SOAP client.
Procedure

1. In the resource tree, locate the desired web service within Desktop > Virtual View Manager Data Services > Web Services, and select the Resource > View WSDL menu option.

2. In the window for providing your log-in credentials, type the user name and password that you used when you logged into the server and opened Virtual View Manager, and click OK.

   The WSDL of the web service is displayed in your browser.

Testing web services

You can type and submit the following URL in your browser to test a web service, substituting appropriate values for the items in italics: http://server_name:9400/services/data service name.html

The server host name is determined by the Base URI setting. When working on the same computer, "localhost" may be substituted. The Base URI configuration setting may also change the published host name.

The default port for web services is 9400, but that can also change by configuration setting.

Test the web service using the Resource menu > Test Service option. The Test Service HTML page clients are special in that they do not invoke the web service directly. They post HTML data to the server, which the server assembles into a SOAP message, issues a SOAP request back to itself, builds an HTML page containing the request and the response, and returns the HTML page to the browser.

Procedure

1. In the resource tree, first select the desired web service from Desktop > Virtual View Manager Data Services > Web Services, and then the Test Service right-click menu option.

2. Enter your log-in credentials, and click OK.

   The web service page is displayed in your browser.

3. Click the desired WSDL operation.

4. Click Execute Operation.

   The web service request and response are displayed as a SOAP message in your browser.
IBM Cognos Virtual View Manager has many security features and design considerations engineered to work together as an integrated system. The security features keep information secure and available for use by only authenticated and authorized individuals with the proper rights and privileges.

This chapter focuses primarily on the user and group rights, resource privileges, and web services security, but here is a brief list of other security features that may not always be apparent when using Virtual View Manager.

**Selected top-level and back-end security features**

IBM Cognos Virtual View Manager has many top-level and back-end security features. Following is a list of these features:

- Passwords sent by JDBC and ODBC to Virtual View Manager are encrypted.
- Passwords passed between Virtual View Manager components are encrypted.
- Passwords in HTTP/SOAP headers for admin functions are encrypted.
- All communication between the components may be encrypted using SSL/HTTPS.
- WSS Web service client security is supported.
- Virtual View Manager to data source SSL is supported with or without WS client authentication where desired and permitted.
- Passwords in metadata are encrypted.
- Passwords for LDAP and dynamic domain users are encrypted or not stored.
- Support for case-sensitive user login for external LDAP is supported.
- Options to include or exclude encrypted user passwords, repository passwords, LDAP, and data source passwords in export files.
- DBA password for the repository is not stored.
- Repository password and the repository connection with Virtual View Manager are encrypted.
- Passwords are not shown in the log files.

The rest of this chapter focuses on the security rights that give users and groups the ability to perform tasks with Virtual View Manager and the privilege settings on resources that secure and provide access to data requested through the Virtual View Manager system. Topics described here include: Resource ownership; ownership reassignment; granting of full privileges; Web Services Security; Transport Layer Security; installation of Certificate Key Stores; Message Level Security; and Pipeline transformations.

- “Rights and privileges” on page 154
- “Rights” on page 154
- “Privileges” on page 159
- “Password changes” on page 166
- “Web services security” on page 167
Rights and privileges

Two managed security layers secure the software and access to data sources. Rights and privileges ensure that only those users with the proper security profile have access to the software and native resources.

Rights give groups and users the ability to perform actions by enabling use of associated tools and options. No rights are given by default to any user except the administrator, who has supreme rights to view and change everything in the system.

Privileges can be given to groups and users so they may view, select, execute, modify, insert, or delete data from defined resources. No privileges are granted by default to newly defined resources except to administrators and the developer (resource owner).

Initially, rights and privileges are assigned as follows:

- The administrator gets all rights and privileges.
- Everybody else has no rights or privileges by default. Rights and privileges must be given by explicit assignment to groups with rights and privileges or alternatively by direct assignment to individual users.

The security architecture provided by rights effectively allows for role-based division of labor and access management by functional group responsibilities including Administration, Development, Operations, Backup, and Restore. End users typically get no rights as they have few reasons to view or change settings or definitions on the server.

The security provided by privileges enables control of data resources so that only those groups and individuals with explicit permissions may use or change data on the data source. Privilege granularity extends down to the level of a column of data in the table, allowing for Read/Write lock down or for full access to columns of data by groups or individuals.

Rights

Rights determine which parts of Virtual View Manager each user can access and use. Rights grant groups and users the ability to administer, develop, and manage the implementation. Rights enable display and modification of server configurations. Other rights enable creation, modification, deletion, and view of defined objects and containers.

Rights are best specified at the level of the group. Users may be members of multiple groups and they inherit and accumulate rights from all the groups to which they belong. By default, administrators have unrestricted rights to change anything within Virtual View Manager.

Group and User Rights templates

When creating new groups and users, the Group Rights and User Rights templates allow for quick assignment of rights based on several user categories.
These user categories are as follows:

- End User
- Developer
- Operations
- Backup
- Restore
- Backup & Restore
- Administrator

The rights granted by the templates increase sequentially from End User with no rights to the Administrator with all rights. You can use these templates as a starting point and change the rights for your purposes:

- **End User template**—Starts with no rights, however the user may request data via ODBC, JDBC, and web service clients. Data is still protected by privileges set at the data source. The end user template does not include rights to use application tools.

- **Developer template**—Grants access to tools and allows view of all status.

- **Operations template**—Grants access to tools, allows read only of server configurations, and allows view of all status.

- **Backup template**—Grants access to tools, allows read of sever configurations, view of all status, read only of resource and user data for backup purposes.

- **Restore template**—Grants access to tools, enables view and modification of sever configurations, all resource and all users.

- **Backup & Restore template**—Provides all rights given by both the Backup and the Restore templates.

- **Administrator template**—Grants complete access and rights to change everything in the system except for some system tables that are locked to ensure proper system functionality. All rights are required in order to gain access to and use Virtual View Manager Administration.

**Group and user rights**

Group rights and user rights are rights assigned at the level of the group or at the level of the individual user. Users who are members of groups inherit all rights assigned to those groups. Each check box gives permissions to use different tools and resources.

Manage enterprise rights at the group level to facilitate role-based management as opposed to assigning rights by individual. Role-based rights management enables more efficient assignment of rights based upon requirements to perform certain tasks.

Managing group rights also prevents problems associated with lack of permissions at initial login for end-users who are dynamically accommodated from an LDAP/iPlanet domain. Virtual View Manager does not introspect new LDAP domains to get lists of potential users because many enterprise directories are very large and user management is slowed and complicated by handling enormous numbers of users who don’t require use of resources. Group management is encouraged and external LDAP groups may be selected to preferentially enable rights and privileges for members of those groups.

Assign the Access Tools right to those developer user groups who should get access to Virtual View Manager tools.
Pre-created users and groups:

The default users—nobody and system—are special users which may not be assigned rights, nor made members of groups.

The cognos/all and dynamic/all groups, and the cognos/anonymous user are pre-created with no rights. Adding rights to them would give rights to all users without proper differentiation.

The anonymous user is not a member of the all group. If a right or privilege is given to the all group, then all authenticated users get that right or privilege, and anonymous users do not receive that permission. Rights and privileges for the anonymous user have to be added explicitly if required for some reason.

Virtual View Manager rights:

The rights that can be granted to a user or group are described in detail in the following sections.

Access Tools right:

Only users granted the Access Tools right may use the Virtual View Manager application or utilize other options and tools, like the API, backup_import and backup_export command line tools.

All Administrators, Developers, IT Operations, and personnel responsible for backup and restoration tasks must have the right to Access Tools. Additional rights must be granted to fully export or import an implemented Virtual View Manager instance.

Without Access Tools, the user can only use JDBC, ODBC and Web Services to access the server and underlying native sources.

All Virtual View Manager tools require the Access Tools right. As this right is almost universally required by all users (with the exception of pure end-users who are interested in the data) that the requirement for the Access Tools right shall be dropped for the sake of brevity when discussing access to other resources. This right allows administrators the ability to deny end users access to Virtual View Manager tools.

Read All Config right:

The Read All Config right grants the user the ability to browse the configuration or by implementation of Admin API. This includes the configuration panels, licenses, and other aspects of the server's configuration.

This right is useful for a Backup role, and may also be of value to developers or operations. The Developer template does not include the Read All Config right by default.

Modify All Config right:

The Modify All Config right grants the user the ability to modify all configurations. Modification to configuration panels, license management, and many other aspects of configuration require the Modify All Config right. The Modify All Config right is powerful and should be granted only to trusted users.
Restoration of a full server backup CAR file requires the Modify All Config right.

Read All Resources right:

The Read All Resources right grants the user the ability to read all resources even if they are not given explicitly given the Read privilege for that resource. Of course, the read privilege should not be confused with the Select privilege which allows for retrieval of the data in the resource. Both the Backup and the Restore rights templates include the Read All Resources right and the right is required for Full Server Backup.

This right would be of value to developers though developers are not granted this right by default with the Developer template.

Read All Resources is just one of many rights required for Manager panel in Studio and Virtual View Manager Configuration access for backup.

Modify All Resources right:

The Modify All Resources right gives the user Grant, Write, Select, Insert, Update, Delete, and Execute privileges on all resources. This right grants the user the ability to modify all resources and all privileges on resources even if they were not explicitly given privileges for that resource. A user with Modify All Resources can change the data source owner and any privileges on that resource.

Note: Administrators may copy and paste resources while maintaining original owner and privilege settings. Users who do not have the Modify All Resources and Modify All Users rights will paste resource copies that do not duplicate owner and privilege settings.

This right is powerful, as it can be used to change data source privileges that may protect sensitive data. This right should be granted only to trusted users.

Read All Status right:

The Read All Status right grants the user the ability to browse the current state and status information. The Active Resource tables are visible to the user with the Read All Status right so that sessions, transactions, requests, caches, data sources, and more are shown on administration panels.

The Read All Status right provides permission to see the logs accessible from the Administration > Server Logs menu option.

This right is useful for developers, operations, and monitoring roles.

Modify All Status right:

The Modify All Status right grants the user the ability to perform actions on the state of the server. This includes terminating sessions, transactions, and requests, clearing caches, and even stopping or restarting the server. Modify All Status right enables testing of all data sources and synchronization of domains.

This right is powerful. It should be granted only to trusted users. Only the Administrator template grants this right by default.

Read All Users right:
The Read All Users right grants the user the ability to browse the list of domains, groups, and users on the server to see how they are configured.

The Read All Users right also allows for Reset of the System Namespace. It does not grant the ability to see any domain or user passwords.

This right is required for full server backup. It is also of value to developers though it is not granted as a default by the developer rights template.

Modify All Users right:

The Modify All Users right grants the user the ability to create and modify domains, groups, and users. It also grants the ability to assign and modify rights for groups and users. Making changes on the Manager - Users pages requires the Modify All Users right. The Modify All Users right effectively gives full administrative powers, as it can be used to grant any other rights.

Unlock Resource right:

The Unlock Resource right gives the user the ability to unlock locked resources.

**Rights summary**

A summary of rights that may be granted to the group and user is shown in the rights summary table.

<table>
<thead>
<tr>
<th>Right</th>
<th>Required for use, view, or access to</th>
<th>Templates where right is present</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access tools</td>
<td>Tools, applications, utilities, and API. This right exists to allow administrators the ability to deny end users access to the tools.</td>
<td>Administrator, Backup&amp;Restore, Restore, Backup, Operations, Developer</td>
</tr>
<tr>
<td>Read All Config</td>
<td>View of configuration settings, licenses.</td>
<td>Administrator, Backup&amp;Restore, Restore, Backup, Operations</td>
</tr>
<tr>
<td>Modify All Config</td>
<td>Change configuration settings, licenses, and other server configuration</td>
<td>Administrator, Backup&amp;Restore, Restore</td>
</tr>
<tr>
<td>Read All Resources</td>
<td>View of all resources, universal Read privilege, Full Server Backup, backup_import, Manager panels, execute any resource procedure, browse and edit resource services</td>
<td>Administrator, Backup&amp;Restore, Restore, Backup</td>
</tr>
<tr>
<td>Modify All Resources</td>
<td>Effective full privileges on all resources, change of privileges on any resource, change owner of a resource, import of privileges and copy/paste of resources with privileges (requires Modify All Users), restore/import (requires almost all other rights to</td>
<td>Administrator, Restore</td>
</tr>
</tbody>
</table>
### Privileges

Privileges indicate who can view, modify, or perform an action on a resource using Virtual View Manager. Privileges such as Read, Write, Execute, Select, Update, Insert, Delete, and Grant are set on folders, data sources, views, and procedures to secure proper access for specified groups and users. Privilege specification provides a comprehensive security layer to safeguard access to containers, folders, and defined objects.

#### Default privileges on a new resource

By default the creator of a folder, container, or resource (the default resource owner) gets all privileges associated with a new object definition. All other users (except for those with admin rights) get no privileges by default for new resources.

Restrictive default initial security settings for newly defined resources protect resources from inadvertent exposure by forcing explicit privilege assignment.

Default Read privileges are given for all example resources, system resources, and parent containers to members of the All Group. Dynamic users who belong to
registered LDAP groups also gain limited Read and Execute privileges to access initially installed example resources and globally available system resources. By default dynamic users and anonymous users are disabled in the installation and must be explicitly allowed by the implementation to enable access.

**Viewing the privileges set on any resource**

Right-select any named resource in the Virtual View Manager resource tree and click Privileges to view the privileges granted to groups and users of that resource.

Some of the access privileges apply to data modeling while others apply to data manipulation. Read, Write, and Grant privileges apply to the resources in the metadata repository whereas Execute, Select, Insert, and Update, and Delete privileges apply to the data in the underlying data source.

The Read and Write privileges are design-time privileges that apply when using Virtual View Manager tools. Modification of an existing resource definition requires user possession of the Write privilege on that resource, whereas simply viewing or incorporating a view as a building block for another view will require Read privileges on the resource. At runtime when using resources contained with resources the Read privilege must be present up the chain in all parent containers for the end-user to exercise runtime privileges on a resource.

The Read privilege on a resource grants the ability to see that the resource exists. The Write privilege on a resource enables modification of the resource definition that defines what and how the native resource may be used.

Run-time resource privileges must be set on the object resource to enable Select or Execute privileges view or use by end-users. Additionally all enclosing containers (parent or folder objects) must give the user Read privileges so that the resource may be accessible.

The Select privilege allows submission of SQL selects to retrieve data.

The Execute privilege enables execution of a procedure.

The Insert, Update, and Delete privileges enable change of table data.

At runtime, we use Read privilege on folders, but Read is not used at all on Tables or Procedures. The intention is that a user with just Select and no Read can select from a table.

**Note:** All users that authenticate using a JDBC/ODBC client of log into Virtual View Manager are automatically members of the All group. Anonymous users are disallowed from access by default configuration setting. LDAP/iPlanet users are non-authenticated dynamic users.

Getting the Read privilege only means that the user may see the resource exists given access to either a client that connects with Virtual View Manager or if the user is assigned Access Tools right then resources are visible with any Virtual View Manager tool.

**Assignment of privileges**

In order to simplify management of both users and resources, developers can assign privileges to appropriately defined groups of users. Role based access control will enable future developers to manage large groups of users without personal knowledge of an individual’s need to know.
Security is best managed at the level of Group permissions. Users inherit all the privileges and rights granted to a group by virtue of being a member of that group. These implicit privileges are shown with a golden check mark.

Impose more finely tuned, granular control by creating additional groups to manage different subsets of rights and privileges from the Virtual View Manager Administration.

The initial privileges view shows groups with both explicitly and implicitly assigned privileges and users with explicitly assigned privileges for that resource. The filter may be narrowed to hide both users and groups without explicit privileges (a green checkmark) or the filter may be removed to show all users and groups.

A custom filter string may also be used on both group and user names. Any string entered in the Filter text box is compared to group and user names using a wildcard at both the front and back end of the string.

If Apply changes recursively to child resources and folders is selected then any privilege changes will propagate down to subordinate or child objects to enable user access to those parts of the selected resource.

**Privilege settings**
You must own a resource and possess the Grant privilege on that resource or have the Modify All Resources right to set, revoke, or change privileges on resource.

Refer to [Ownership of resources and the grant privilege](#) on page 166 for more information.

Manage group privileges like group rights so that groups of end-users will have appropriate Read access to all parent resources according to appropriate need. Users inherit all the rights and privileges given to groups in which they are members.

Assign specific resource privileges to end-users groups who will be using JDBC, ODBC, or web service clients. Group management of privileges is especially important because individual LDAP/iPlanet users will not be available for management by individual exception, until after that user has either logged into an application, or used a JDBC, ODBC, or web service client interface.

Every user defined in Virtual View Manager gets the most basic right to Read all default installed resources. LDAP/iPlanet dynamically assigned users get a similar subset of that basic privilege as outlined in the previous graphic.

Of course, individual users may be assigned rights and privileges for any resource by direct assignment, but only after they appear in the system. Virtual View Manager does not introspect new LDAP domains to get lists of potential users because many enterprise directories are enormously large and full of users who may not require use of resources. Appropriate privileges can be assigned to groups of users.

**Note:** End-users may experience difficulties with resource availability and insufficient access privileges if they do not have necessary parent container Read privileges. End-users must have Read privileges for each container from the root all the way to the leaf resource to enable client view and use a resource.
Developers and resource owners should make sure that appropriate groups and users get Read privileges on all parent folders so that contained resources may be both visible and accessible.

Setting privileges on a resource:

Follow these steps to set the privileges on a resource.

Procedure

1. Right-click the resource name and select Privileges, or select the resource and choose Resource > Privileges.

   The Privileges panel displays your privileges and those of all groups and users with explicit privileges by default. It is best to set privileges by groups, but if you wish to explicitly set privileges for users who are not shown, then toggle on the Show All radio button.

2. To set design-time privileges, select the Read and Write boxes, keeping these things in mind:
   - Make sure that groups and users given new privileges also get the Read privilege on all parent resources in the path of the current resource.
   - The Read privilege enables view of the object design, projections, schema, SQL, and annotation
   - The Read and Select privileges enable the group or user to view dependencies and the execution plan. These users will be able to run the SQL of the view.
   - Users with the Read, Write, and Select privileges will be able to save, cache, and publish the view.

3. To set runtime privileges, check the appropriate privileges to enable Execute, Select, Update, Insert, or Delete.

   Refer to "Privilege enforcement on sources underlying a resource" on page 164 to know how to grant privileges to a source on which a resource depends.

Containers and leaves

For the sake of understanding access privileges, resources may be broadly classified as containers and leaves or parent objects and child objects.

A container is a resource that contains other resources, which in turn can be containers or leaves. A leaf is resource which does not contain any other resource.

Containers may be referred to as parents, and a resource that has a parent is a child object. By this convention, any resource except the root in the resource tree is a child of some parent.

Privilege enforcement on containers and leaves:

When you designate a privilege for groups or individual users of a container, that privilege may be propagated down to all child objects within that container using the Apply changes recursively to child resources and folders check box.

To open up an entire parent directory for view and use by all users, select the top-most node in the resource tree or on the parent container, open the privileges dialog and set the READ privilege for the container. Select the Apply changes recursively to child resources and folders check box to propagate the Read privilege down to all child objects within that directory.
The following table describes how privileges are set on containers and leaves, and what can and cannot be done with those privileges.

In the table:
- X means that the privilege is granted on the resource.
- — means that the privilege is not granted on the resource.
- N/A means that the privileges is not applicable to the resource.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Design-Time Privileges</th>
<th>Run-Time Privileges</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
<td>Execute</td>
</tr>
<tr>
<td>Leaf</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Leaf</td>
<td>X</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Leaf</td>
<td>X</td>
<td>X</td>
<td>—</td>
</tr>
<tr>
<td>Leaf</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>Leaf (views with parameters, SQL Script procedures, Java procedures, and web service operations)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Container</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

All the parent resources in the path of the current resource must have the Read permission.
Table 17. Settings for privileges on containers and leaves (continued)

<table>
<thead>
<tr>
<th>Resource</th>
<th>Design-Time Privileges</th>
<th>Run-Time Privileges</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
<td>Execute</td>
</tr>
<tr>
<td>Container</td>
<td>X</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Container</td>
<td>X</td>
<td>X</td>
<td>—</td>
</tr>
</tbody>
</table>

Privilege enforcement on sources underlying a resource:

When a resource depends upon or uses another resource internally, the user who requests view of the resource should have privileges to get data from those dependencies or underlying resources.

For example, if a user wants to see the data of a complex view named View_A, where View_A executes Procedure_B (which draws on data from physical source Table_D) and also selects from physical source Table_C. For that user to successfully perform a SELECT on View_A, the following privileges:

<table>
<thead>
<tr>
<th>Privilege to grant</th>
<th>On the resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>All of View_A’s parents</td>
</tr>
<tr>
<td>Select</td>
<td>View_A</td>
</tr>
<tr>
<td>Read</td>
<td>All of Procedure_B’s parents</td>
</tr>
<tr>
<td>Execute</td>
<td>Procedure_B</td>
</tr>
<tr>
<td>Read</td>
<td>All of Table_C’s parents</td>
</tr>
<tr>
<td>Select</td>
<td>Table_C</td>
</tr>
<tr>
<td>Read</td>
<td>All of Table_D’s parents</td>
</tr>
<tr>
<td>Read</td>
<td>Table_D</td>
</tr>
</tbody>
</table>

Display privileges:
By default, every object resource defined in Virtual View Manager is created with full privileges for the object creator/developer and with no privileges for anyone else, except for administrative users who have either the Modify All Resources or the Modify All Users rights.

Review and modify access privileges to read, write, or grant other users and groups privileges to use object resources. Virtual View Manager allows you to grant privileges or revoke them. You can launch this window by right-clicking the desired resource name and selecting Privileges.

When a privilege is granted to a user, the corresponding privilege check box is selected in the Privileges panel.

- All privileges (Read, Write, Execute, Select, Update, Insert, and Delete) can be granted on any container although only Read and Write are relevant to a container.
- For leaf resources, the Select/Execute are synonymous, and are mutually exclusive. If the leaf resource is a view, the privilege to be granted is Select and if the leaf resource is a procedure the privilege to be granted is Execute.

The Properties window for a published resource also displays the privileges on the original source.

Privileges granted on either the published resource or the original source give the user or group equivalent powers to view, modify, execute, or select on the resource.

**Privilege granularity:**

The security architecture for privileges provides granular information control to the resource owner who can lock down the data source and grant privileges to groups or individual users enabling them to see as much, or as little as may be required.

Administrators with the Read All Resources right may see all resource definitions and associated privileges. With the Modify All Resources rights, the admin may unlock resources by assigning new privileges to other groups and users or restrict access by deselecting privileges if required.

Privileges may be assigned for an entire domain, selected groups, or individual users. Privileges may be set for an entire folder and for all child objects within that parent recursively with a single click. Conversely, privileges enable more granular control to restrict access to, and change of data down to level of the individual table column.

As stated previously, users must have Read privileges on a parent container to view and utilize resources contained within the parent container. In other words, child object resources are not available or even visible to users, without read privileges on all pertinent parent containers regardless of the extent of privileges on a subordinate object.

Granular control to access, select, view, and change any data source object with group rights and privileges.

Resource owners and administrators can assign differentiated access to data sources, tables, columns, web services, methods, and procedures for groups and
users. For LDAP domains, groups can be granted privileges at the same time, since individual members of the domain do not appear until they have logged into the system for the first time.

Ownership of resources and the grant privilege
Each resource has an owner. The user who creates a resource is the default owner initially. An owner of a resource automatically has all privileges on that resource. Administrators with the Modify All Resources right may change the owner of a resource.

The resource owner (or a delegate given the Grant privilege on that resource) defines privileges for groups and users who may need to view, access, and use a resource. Privileges may be defined for a parent object in the directory and they may be applied to child resources and sub-folders recursively with selection of the check box at the bottom of the privileges window for that resource.

Initially only the owner of a resource or an administrator can grant specific access privileges to any groups or user for a resource. Privileges may be revoked just as easily. Other developers, who are not administrators or owners of the resource, may view privileges set on a resource, but they will not be able to change those privileges unless they have been given the Grant privilege on that resource or they have the Modify All Resources right. The Modify All Resources right enables change of any privilege settings.

The owner or administrator has the option to grant or revoke privileges on the owned resource at any time. The owner may even revoke his own privileges (for example, to proactively prevent accidental deletion of data), and later re-grant those privileges as necessary.

An administrator with the Modify All Resources right can change privileges on all resources that they have access to view, but that administrator isn’t necessarily empowered to read all resources.

A user who is given ownership of a resource (by being the original resource creator, or by assignment) may share all the privileges of ownership, by giving the Grant privilege to other users or groups.

Grant privilege
The Grant privilege enables another user or group of users the ability to set privileges for that resource. The Grant privilege effectively empowers other users with the same abilities as the original resource owner. Users with the Grant privilege on a resource are essentially co-owners of a resource who may give or take away privileges from other users.

Both the administrator and the owner of a resource have the ability to change the privileges associated with a resource and subordinate objects, but even the administrator may be restricted from modifying all resources so that object ownership rights to grant privileges may be controlled solely by the object creator.

Password changes
Administrators can change any Virtual View Manager domain passwords for users managed by that domain. Users defined by the Virtual View Manager domain can also change their own password.
Changing your own password

Follow this step to change your password.

Procedure

Select File > Change Password.
Administrators with the Modify All Users right may use Virtual View Manager Administration to change the password of any Virtual View Manager domain user.

Changing passwords for other domain users

Follow these steps to change the password for other domain users.

Procedure

1. Click Manager > Users > User Management.
2. Select the user, and enter the new password.
3. Click OK.

Web services security

Virtual View Manager offers WS-security at two levels: transport layer level and message level. Secure web services published by Virtual View Manager using Web Services Security (WSS). Virtual View Manager enables signing of messages with digital certificates to identify the source of the message and to encrypt the message for secure delivery to the end-user client.

Supported WS-security standards

Virtual View Manager supports several WS-Security standards.

The following standards are supported:

- Username Token profile V1.0 (see: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf)
- X.509 Token Profile V1.0 (see: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf)

Virtual View Manager allows for use of SAML and Kerberos, and certificate formats such as X.509 to remove the need to encode keys and message signatures into ASCII before sending.

WS-Security incorporates security features in the header of a SOAP message and works in the application layer, ensuring end-to-end security.

Web services need security for the SOAP message that is carried from one end to another.

Transport Layer Security (TLS) is supported to ensure message integrity and confidentiality via HTTPS reducing performance overhead. If the messaging needs to pass through a proxy server, however, then TLS should not be used without special handling considerations.
**Transport layer security (TLS)**

Transport layer security can be used to protect Web service message content from being read or modified by third parties during transport.

Transport layer security encrypts TCP/IP packets, and does not encrypt the data. The data are in clear at each end-point. This level of security is sufficient if the data transfer between a client and server is direct, meaning there is no intermediate entity between the client and the server. Thus the two end-points would be A and C, where A represents the server and C, the client. If there is an intermediate source, say B between A and C, C cannot trust that data sent from A were modified or not by B.

Security at the transport layer level is provided for WSDL data sources and Web services which are available to external client applications.

**Securing a web service port**

Follow these steps to secure a web service port.

**Procedure**

1. Open the desired port to configure it.
2. In the section Transport Level Security, select None or SSL/TLS.
3. In the Authentication section, select the method of client authentication.
   - **Basic Authentication**—allows a web browser or other client program, to provide credentials in the form of a user name and password when making a request.
   - **WS-Security Username Token**—a security token represents a collection of claims where a claim is a declaration that an entity makes (for example, name, identity, key, group, privilege, capability, and so on).
     There are two kinds of tokens: Unsigned Security Token, such as a username, and Signed Security Token, such as x.509 certificates or Kerberos tickets.
4. Indicate whether client programs can use all or any of the authentication method(s) selected in the preceding step.
5. Save the configuration.

**Virtual View Manager data service security**

Secure data services and the Web services they contain with digital certificates that verify the identity of your servers and encrypt the data messages passed to and from your clients.

Before proceeding further, have a certificate keystore file ready. You can use one of two types of supported certificate keystores: Java Key Store or PKCS12. Virtual View Manager provides a self-signed certificate to support HTTPS which should be replaced at your earliest convenience. The digital certificate may be issued by a certification authority (CA) or you may generate your own self-signed certificate given certain drawbacks of failing to present a trusted third-party certifier.

**Securing a WSDL data source web service with digital certificates**

Follow these steps to secure a WSDL data source web service with digital certificates.

**Procedure**

1. Open the data service that you want to secure.
2. Click the **Import Certificate Key Store from** file button to import your certificate keystore file(s).
3. Enter the path to the keystore.
4. Specify the keystore type (jks for a Java Key Store or pkcs12 for PKCS12).
5. Enter the password for the key store.
6. Click **OK**.
7. The window displays a list of certificates available.
8. When the server identity is challenged to connect with this data source, the server will cycle through every certificate until access to the data source is granted. The certificates are not associated with any specific user, and there is no pass through behavior.
9. Make a note of the **Alias** for each keystore.
10. You will need to provide such alias if you want to provide a message level pipeline security, when you encrypt an element or sign an element.

**Message level security**

If the messaging is indirect, meaning that there is an intermediate source in the transport layer, we need what is called message level security.

For indirect messaging, such as when there is more than one receiver or a sender of a message, the message needs to be secured at different levels if the integrity of data needs to be intact. Suppose if there are three parties—A, B, and C—involving in transferring a message (M) from one party to another, each party (sender/receiver) may need to identify itself in order to be recognized by one at the other end and to ensure that data integrity was not affected during the transport of the message.

**Message pipeline steps**

A pipeline defines multiple instructions that are processed simultaneously.

You can provide message level security by defining multiple steps of well-defined pipeline processing on request and response messages. Specifically, you can provide message level security at the following locations of a Web service or WSDL data source:

- Web service operation or published resource
- WSDL operation or published resource
  - Port-level security settings apply to all the operations within that port.
  - Operation-level security settings for a given operation override the security settings of the port.

**Viewing the available pipeline:**

Follow this step to view the available pipeline.

**Procedure**

Open the desired operations folder or a particular web service that is to be secured.
Message layer security is provided for the web service at the operations level.
In the editor that opens on the right, the **Message Pipeline** panel opens.
The **Request Message Pipeline** and **Response Message Pipeline** sections allow for message processing steps to be added to the pipeline.
Except for the Custom step, each of the pipeline steps corresponds to a system built-in procedure available at /lib/services/ in the server. The following sections describe how to create each of these pipeline steps.

**Pipeline step: log message to file:**

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

This pipeline step writes the contents of a SOAP message to a file in the specified path. You can create this pipeline step for a request message and/or for a response message.

The process of creating this pipeline step is described first so that the outputs from other pipeline steps can be written to the log file created by this step.

*Creating the log message to file pipeline step:*

Follow these steps to create the log message to file pipeline step.

**Procedure**

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:
   - Click the Browse button to specify an existing file.
   - If the file does not exist, supply the fully qualified path and file name, such as C:\pipelineLog.txt, to create the file.
4. In the File Mode field, specify how the message should be logged.
   - **APPEND**—Selecting this option would append the message to the end of the file.
   - **OVERWRITE**—Selecting this option would overwrite the file with the contents of the message.
5. (optional) In the Header field, supply a header note which will be written to the file immediately before the message contents. This value can be null.
6. (optional) In the Footer field, supply a footer note which will be written to the file immediately following the message contents. This value can be null.
7. In the Pretty Print drop-down list, select true if you want the message to be formatted with indents, and false if you do not want the message to be formatted.
8. Click OK, and save the step.

The output of this pipeline step is the modified XML document or element. A sample output is reproduced here in part.

**Pipeline step: add username token:**

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

You can use this pipeline step to authenticate a message. If the server hosting the Web service requires that the request message has a username token, then you use this pipeline step.
This pipeline step will add a **WS-Security UsernameToken** to a SOAP envelope. The **UsernameToken** will be 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, the header will be created.

Creating the add username token pipeline step:

Follow these steps to create the add username token pipeline step.

**Procedure**

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**.
   
   Values for the following fields need to be supplied in the **Add Username Token** window:
   
   - **Actor**—Specifies the SOAP Actor attribute. The Actor attribute can be null, which would indicate that the recipient is the ultimate destination of the SOAP message.
   
   The SOAP actor attribute may be used to address the Header element to a particular endpoint. A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path. Not all parts of the SOAP message may be intended for the ultimate endpoint of the SOAP message but, instead, may be intended for one or more of the endpoints on the message path.

   The value of the Actor attribute is a URI, which can be used to specify the recipient of a header element.

   - **Must Understand**—Indicates whether a header entry is mandatory or optional for the recipient to process.

   - **Username**—Valid user name to access the Web service server.

   - **Password**—Valid password associated with Username.

   - **Password Type**—Determines how the password will be encoded in the Username Token:
     - **DIGEST**—Password will be rendered in a digested text in the message
     - **TEXT**—Password will be rendered in clear text in the message

3. In the **Actor** field, type a Uniform Resource Identifier (URI).
4. In the **Must Understand** drop-down list, select true or false.
   
   **True** 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.

   **False** indicates that the receiver processing the Header need not recognize the element.

5. Enter a valid user name to access the server in the **Username** field.
6. Enter a valid password associated with the entry in the **Username** field.
7. Specify the password type—**DIGEST** or **TEXT**—in the **Password Type** drop-down list.
8. Click **OK** after supplying all the required information, and save the step.

**Pipeline step: create element:**
This pipeline step corresponds to the system procedure CreateElement, which is available in /lib/services/.

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

Creating the create element pipeline step:

Follow these steps to create the create element pipeline step.

Procedure

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. Enter the fully qualified name of the element in the Element Name field.
   This value may not 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 0 (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 may not 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. Click OK after supplying all the required information.

Pipeline step: custom

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:

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

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

Notice that the first parameter is an input parameter of type XML.

Creating the custom pipeline step:

Follow these steps to create the custom pipeline step.
Procedure
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 Custom.
3. Use the Browse button to locate and select the desired custom procedure.
4. Supply the input parameter value in the Parameter Values field, and click OK.

Pipeline step: delete element
This pipeline step corresponds to the system procedure DeleteElement, which is available in /lib/services/.

This pipeline step deletes one or more nodes from an XML document or 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 as described in “Pipeline step: create element” on page 171

Creating the delete element step:
Follow these steps to create the delete element step.

Procedure
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 Delete Element.
3. Type the path to the parent of the element in the Parent XPath field.
   This path will be used to select the nodes to be deleted. The path will be evaluated against the root node. All resulting nodes will be deleted. This entry should not contain any namespaces.
4. In the Prefix field, specify the namespace prefix used in the Parent XPath field.
5. In the Namespace field, specify the namespace URIs used in the Parent XPath expression.
6. Click OK after supplying all the required information, and save the step.

Pipeline step: encrypt element
This pipeline step corresponds to the system procedure EncryptElement, which is available in /lib/services/.

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

Creating the encrypt element pipeline step:
Follow these steps to create the encrypt element pipeline step.

Procedure
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 Encrypt Element.
   For details on Actor and Must Understand, see “Creating the add username token pipeline step” on page 171
**Element Name**—Specifies which element in the message should be encrypted. Typically this procedure will be used to encrypt either the body or header of the SOAP message.

**Encryption Algorithm**—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 will require an unrestricted Java Cryptography Extension (JCE) policy file to be installed in the server's JVM.

**Certificate Alias**—The alias of a certificate or public key in the key store that will be used to encrypt the symmetric key that is used to encrypt the element. It may not 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 may be used as the Certificate Alias in a pipeline step. For further details, see “Securing a WSDL data source web service with [digital certificates](#)” on page 168.

3. In the **Actor** field, type an URI.

4. In the **Must Understand** drop-down list, select true or false.

5. In the **Element Name** field, type the name of the element in the envelope to encrypt.
   
   It may be null. If null, the element name will default to:

   ```
   {http://schemas.xmlsoap.org/soap/envelope/}Body
   ```

6. In the **Encryption Algorithm** field, 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.

7. In the **Certificate Alias** drop-down list, select 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 will be listed here. Otherwise, you need to import the certificates as described in “Securing a WSDL data source web service with [digital certificates](#)” on page 168.

8. Click **OK** after supplying all the required values.

**Pipeline step: process security header**

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

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 will be processed.
- All security elements in the header will be evaluated.
- If any header security elements indicate that the envelope contains signed elements, the signatures of those elements will be verified.
- If any header security elements indicate that the envelope contains encrypted elements, those encrypted elements will be decrypted.

**Creating the process security header pipeline step:**

Follow these steps to create the process security header pipeline step.

**Procedure**

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

3. Supply a value in the Actor field, if necessary, and click OK.

4. Save the step.

**Pipeline step: set environment from node value**

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

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

**Creating the set environment from node pipeline step:**

Follow these steps to create the set environment from node pipeline step.

**Procedure**

1. In the pipeline editor for the resource that is to be secured, click the Add button ( + ) in the Request Message Pipeline section, and select Environment From Node. The XPath entry will be evaluated to some text which will be stored in the variable name provided in the Variable Name field.

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

2. Type the XPath value in the XPath field.

3. Type the desired variable name in the Variable Name field, without a space in the name string.

4. Type the namespace prefixes and their corresponding namespace values in the Prefix and Namespace fields.

5. Click OK.

**Pipeline step: set node from environment**

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

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.

**Creating the set node from environment pipeline step:**

Follow these steps to create the set node from environment pipeline step.

**Procedure**

1. In the pipeline editor for the resource that is to be secured, click the Add button ( + ) in the Request Message Pipeline section, and select Set Node From Environment. The XPath expression will be evaluated to an element.
Variable Name is the name of an environment variable. It is an arbitrary string, and is not case-sensitive. Both sample and SAMPLE are considered the same.

2. Type the XPath value in the XPath field.
3. Type the desired variable name in the Variable Name field, without a space in the name string.
4. Type the namespace prefixes and their corresponding namespace values in the Prefix and Namespace fields.
5. Click OK, and save the step.

Pipeline step: sign element
This pipeline step corresponds to the system procedure SignElement which is available in /lib/services/.

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.

Creating the sign element pipeline step:
Follow these steps to create the sign element pipeline step.

Procedure
1. In the pipeline editor for the resource that is to be secured, click the Add button ( ) in the Request Message Pipeline section, and select Sign Element.
   For details on Actor, Must Understand, Element Name, and Certificate Alias, see “Creating the encrypt element pipeline step” on page 173.
2. Click OK after supplying all the required values.
Chapter 10. Performance tuning

This chapter describes the tasks that are related to fine-tuning your query execution to enhance server performance.

Specifying the properties of a join

One of the tasks you might perform when you design a view is to join tables. You can sometimes refine the properties of a join to enhance query execution performance.

You can refine join properties by
- specifying how the joined columns should be compared
- specifying the join algorithm and cardinality
- choosing the semi-join optimization
- specifying whether to include all the rows from a specific table

Procedure
1. Join the tables as described in “Adding resources to the Model panel” on page 82.
2. Right-click the desired join line, and select Properties, or double-click the join diamond or line.
3. In the Join Properties window, specify how the columns should be compared by selecting the desired operator from the upper drop-down list.
4. In the Include rows section:
   - Select the upper box to specify the LEFT OUTER JOIN.
   - Select the lower box to specify the RIGHT OUTER JOIN.
5. In the Join Details section:
   - In the Specify Join Algorithm drop-down list, select the algorithm to use for the join.
   - Specify the Left Cardinality constraint. Provides cardinality hint for the left-hand side of a join. It should be a positive numerical value, for example 50.
   - Specify the Right Cardinality constraint. Provides cardinality hint for the right-hand side of a join. It should be a positive numerical value, for example 500.
   - Select Semijoin Optimization to minimize the number of rows to be processed for the join, and the query engine’s performance is highly enhanced.
   - Select one of the following order options:
     Default Ordering—applies default ordering
     Swap Order—swaps the left and right sides of a join
     Force Join Ordering—join ordering optimization will not be applied
6. Click the SQL tab to verify the SQL, as in the following example:

   ```sql
   SELECT products.ProductID, orderdetails.UnitPrice, orderdetails.Status
   ```
FROM /shared/examples/ds_orders/orderdetails orderdetails
RIGHT OUTER (OPTION SWAP_ORDER="True") JOIN /shared/examples/ds_orders/products products
ON orderdetails.ProductID = products.ProductID
LEFT OUTER (OPTION FORCE_ORDER="True") JOIN /shared/examples/ds_orders/orders orders
ON orderdetails.OrderID = orders.OrderID

7. Click OK.

Join algorithms
IBM Cognos Virtual View Manager uses several algorithms and the SEMIJOIN optimization to execute joins on table columns.

The Join Properties editor offers the following join algorithm options—AUTOMATIC, HASH, NESTEDLOOP, and SORTMERGE—and the Semijoin Optimization option, all of which are described here. If you don’t select any of these algorithms, the Virtual View Manager query engine will attempt to find the best algorithm and join optimization to use for your SQL.

Automatic option
By default, Virtual View Manager automatically optimizes the SQL execution plans for any query, given estimates of left and right cardinality or other statistical information. Automatic is not a join algorithm or a join optimization, but it is offered as a selection to allow the query engine to optimize based on an analysis of the SQL with known database statistics.

The SQL execution plan may be specified explicitly with one of the other options (refer to “Specifying the properties of a join” on page 177 or the Reference Manual section on SQL Keywords), however specification of a join option only strongly influences what join algorithm or optimization is first attempted, not necessarily that which is ultimately used. If the option specification is incompatible with the data sources or incorrectly specified for other reasons, then the execution plan may 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. If you are joining two tables, the optimizer would use the inner table to build a hash table on the join key in memory. It then would scan the outer table, probing the hash table to find the joined rows.

The HASH algorithm only requires a single pass of each table. It is best used when the inner table fits in available memory. The cost is then limited to a single read pass over the data for the two tables.

However, if the hash table grows too big to fit into the memory, then the optimizer would break it up into different partitions. As the partitions exceed allocated memory, parts are written to temporary segments on disk.

Note: If the process is expected to write to disk a performance advantage may be obtained by directly forcing the process to write to disk from the beginning to avoid costly read/write cycles and excessive memory usage. Refer to the Query Engine Options section on "FORCE_DISK" in the Reference Manual.
**Nestedloop join option**

The NESTEDLOOP algorithm can be used without any data structure information, but it is inherently not optimized for performance. Specifically, NESTEDLOOP is useful when small subsets of data are joined and if the join condition is an efficient way of accessing the right table.

When the outer (driving) table has numerous rows that force multiple probes of the inner table, the NESTEDLOOP algorithm will become costly to run. It will become even more costly if the inner table is expensive to access.

**Sortmerge join option**

The SORTMERGE algorithm is a streaming operator. If both of the input streams may be ordered, the join operator works efficiently and quickly in a small memory footprint.

The SORTMERGE also has additional conditions such as both sides of the join must be ordered. Ordering has to be compatible with the join criteria.

For example consider the following join criteria:

\[ \text{ON T1.A = T2.X AND T1.B = T2.Y} \]

In order for the SORTMERGE join to be performed, the left side has to be ordered by A ASC, B ASC. Similarly, the right side has to be ordered by X ASC, Y ASC. If a side is not ordered or does not have compatible ordering, the query engine will automatically check to see whether a compatible ORDER BY may be pushed to the data source so that a SORTMERGE join may be performed.

SORTMERGE is not compatible with the SQL when ORDER BY can't be pushed to the data sources. Ordering must be performed on both sides for a SORTMERGE join.

To preempt the query engine from choosing SORTMERGE over HASH, specify the value of the SORTMERGE option to be false, as follows:

\[ \text{option SORTMERGE="false"}. \]

**Semijoin optimization option**

A semijoin is a very fast join optimization that 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).
Note: The semijoin can only be attempted if the right hand side may be queried as a node which fetches against a data source that supports an IN or an OR clause. Table cardinalities must also be supplied or known for this optimization.

About specifying SEMIJOIN manually:

The SEMIJOIN optimization may be specified manually in the SQL specified by the data source. However, even if the SEMIJOIN optimization is not specified explicitly as an option in the SQL, the query engine optimizer still attempts to apply it in the execution plan if it may enhance performance.

SEMIJOIN usage: SELECT column1 FROM table1 INNER \{OPTION SEMIJOIN\} JOIN table2 ON table1.id = table2.id

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.

Mechanism and restrictions:

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 left side which is then added to the SQL sent to the RHS.

A restriction on a SEMIJOIN exists because various database 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, then the query engine creates an execution plan that attempts a partitioned semijoin, where the IN list is broken up into chunks of 100 or fewer unique values and multiple queries are executed against the right hand source. If the cardinality is still too large, the system will fall back to the HASH algorithm.
Another restriction is set to keep the LHS from exceeding a predetermined limit that would inordinately burden the join. The LHS cardinality restriction is a configurable setting that can be used to enlarge or restrict the number of rows that will trigger an automatic semijoin. You can change this setting in Virtual View Manager.

*Changing the left hand side cardinality setting for semijoins:*

Follow these steps to change the left hand side cardinality setting.

**Procedure**
1. Click **Administration** > **Configuration**.
2. In the **Configuration** window, click **Virtual View Manager Server** > **SQL Engine** > **Optimizations** > **Max Left Cardinality for Automatic Semi-Join Selection**.
   Change the setting to either enlarge or reduce the maximum number of rows in the left hand side of a join that will trigger an automatic semijoin.

**Equi and Non-Equi Semijoins with known table cardinalities:**

When table cardinalities have been gathered or are known, the SEMIJOIN can be used with conditions that join tables on either an equality condition or with a nonequi (non-equality) condition like greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=), and not equal to (<>). Non-equi semijoin optimizations require that table cardinality be known (or specified) or the query will be forced to use a less than optimal merge strategy.

**Enforcing join ordering**

It is usually best to allow the query engine to reorder the query execution plan based upon statistics gathered from the data source table. However, given knowledge of the table contents, it is often advantageous to force the processing order derived from the written SQL.

In this case, you can direct the query engine to follow the order in which you joined the tables.

**Procedure**
1. Join the tables in the order you want.
   For more information, see Chapter 4, “Views,” on page 75.
2. Right-click the desired join line and select **Properties**.
3. In the **Join Properties** window, select **Force Join Ordering**.
   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.

**Types of joins**

Inner and outer joins may be either Equi or Non-Equi. Most table joins use an equality condition, an Equi join, comparing columns and rows with like values. For anything other than an equality join, only the nested loop algorithm is used.

Left, right, Virtual View Manager supports the following types of joins: Equi-Join, Non-Equi-Join, left/right outer join, and full outer join.
The following table lists algorithms used and the types of joins that they may be used with, where an X indicates algorithm support.

Table 18. List of algorithms and join types

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Equi-Inner Join</th>
<th>Non-Equi Join</th>
<th>Left/Right Outer Join</th>
<th>Full Outer Join</th>
<th>Semi-Join Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Nested Loop</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sort Merge</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displaying SQL execution plans for views

The SQL execution plan is a tool that determines how the server executes the SQL for a particular view.

Procedure

1. Open the view in IBM Cognos Virtual View Manager.
2. Click the **Show Execution Plan** button on the editor toolbar.

   The execution plan for the view is displayed in the lower pane.
   In the execution plan, the server transforms the view query into a tree of processing nodes through which data from the appropriate data sources flow and become processed into the desired result.
   The **FETCH** node displayed in the left pane represents the rows fetched from the data source **ALL_DATASOURCES**. The fetched rows then flow through an **ORDER BY** node which orders the rows by the column **DATASOURCE_NAME**. Subsequently, the ordered rows flow through the **SELECT** node which collects all the values before returning the values to the client.

   Plans for queries that call procedures in the FROM clause show the details of what the procedures do.

Execution plan contents

The following table lists all the possible nodes that can display in the left pane of the **Execution Plan** window.

Table 19. List of nodes and functionality available from the left pane of the Execution Plan window

<table>
<thead>
<tr>
<th>Node</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGGREGATION</td>
<td>Shows plans for aggregate functions that are not part of a GROUP BY clause.</td>
</tr>
<tr>
<td>CROSS JOIN</td>
<td>Merges two streams of incoming rows and produces one stream of rows that is the cartesian product of the two streams.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Removes all incoming duplicate rows.</td>
</tr>
<tr>
<td>FETCH</td>
<td>Produces the rows resulting in the execution of a query on a particular data source.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Removes the incoming rows based on a criterion. Specifically, only rows that satisfy the criterion are passed on through.</td>
</tr>
</tbody>
</table>
Table 19. List of nodes and functionality available from the left pane of the Execution Plan window (continued)

<table>
<thead>
<tr>
<th>Node</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION</td>
<td>Shows how a function is executed.</td>
</tr>
<tr>
<td>FULL OUTER JOIN</td>
<td>Merges two streams of incoming rows and produces one stream containing the SQL FULL OUTER JOIN of both streams.</td>
</tr>
<tr>
<td></td>
<td>Refer to a SQL reference manual for a description of FULL OUTER JOIN.</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Reorders the incoming rows so that they are grouped by a criterion.</td>
</tr>
<tr>
<td></td>
<td>For example, if the rows are grouped by a name, all rows with the same name will be combined into a single row.</td>
</tr>
<tr>
<td>IN</td>
<td>In subquery nodes for un-pushed IN subqueries.</td>
</tr>
<tr>
<td>JOIN</td>
<td>Merges two streams of incoming rows and produces one stream containing rows that satisfy a criterion that applies to both streams.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Reorders the incoming rows to satisfy a sorting criterion.</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>Produces rows resulting in the execution of a query or stored procedure call on a particular data source.</td>
</tr>
<tr>
<td>RIGHT OUTER JOIN</td>
<td>Merges two streams of incoming rows and produces one stream containing the SQL right outer join of both streams.</td>
</tr>
<tr>
<td></td>
<td>Refer to a SQL reference manual for a description of RIGHT OUTER JOIN.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Applies functions on the column values on the rows. This node produces exactly the same number of rows that it reads.</td>
</tr>
<tr>
<td>UNION</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table lists all the possible nodes that can display in the right pane of the Execution Plan window.

Table 20. List of nodes and functionality available from the right pane of the Execution Plan window

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Shows the name of the algorithm used by the node. These names are only meaningful to the support staff when diagnosing an issue.</td>
</tr>
</tbody>
</table>
Table 20. List of nodes and functionality available from the right pane of the Execution Plan window (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Data Source Read Time</td>
<td>Time spent by background threads in all FETCH and PROCEDURE nodes in the execution plan.</td>
</tr>
<tr>
<td>Background Node Processing Time</td>
<td>Time spent processing this node by a background thread.</td>
</tr>
<tr>
<td>Background Server Processing Time</td>
<td>Time spent by background threads in all the nodes (except for FETCH and PROCEDURE) in the execution plan.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Shows the criteria used by a JOIN, FILTER, GROUP BY, or ORDER BY node.</td>
</tr>
<tr>
<td>Elapsed Execution Time</td>
<td>Amount of wall-clock time that the server used to execute the query. This time is the total of Query Initialization Time, Foreground Server Processing Time, and Foreground Data Source Read Time.</td>
</tr>
<tr>
<td>Foreground Data Source Read Time</td>
<td>Fraction of the Elapsed Execution Time that the server used to read data from the data sources. By comparing this time with Foreground Server Processing Time, you can determine how much time was spent by the server vs. the time spent in the data sources.</td>
</tr>
<tr>
<td>Foreground Node Processing Time</td>
<td>Fraction of the elapsed time used by the node. This time can be zero if the node was processed by a background thread.</td>
</tr>
<tr>
<td>Foreground Server Processing Time</td>
<td>Fraction of the Elapsed Execution Time that the server used to execute the query. This time comprises 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, you can determine how much time was spent by the server vs. the time spent in the data sources.</td>
</tr>
<tr>
<td>Projection</td>
<td>Shows the set of output columns in a SELECT node.</td>
</tr>
<tr>
<td>Query Initialization Time</td>
<td>Time the server used to create and optimize the execution plan.</td>
</tr>
<tr>
<td>Rows Returned</td>
<td>Indicates the number of rows produced by a 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.</td>
</tr>
<tr>
<td>Speed Up due to Concurrency</td>
<td>Estimate of how much faster the query ran because of threading. If the value is 0%, there was no speed up. If the value is 100%, the query would have taken twice as long without threading.</td>
</tr>
<tr>
<td>SQL</td>
<td>Shows the query's SQL. In a FETCH or PROCEDURE node, this field contains the actual data-source-specific query that is sent to the data source.</td>
</tr>
</tbody>
</table>
**About updating query execution plans**

You can update the query execution plan for a view.

The **Execution Plan** panel has a toolbar button (Generate Plan) and an input field (Refresh Rate) for your use to update the query execution plan.

Clicking the **Generate Plan** button forces the query engine to update the execution plan immediately.

**Getting query execution statistics:**

You can view how much data was processed by a node.

The **Execution Plan** tool lets you execute a query and determine how much data was processed by each node and which nodes took the most time.

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.

In the **Refresh Rate** input field, you can specify the regular interval at which the execution plan needs to be updated.

Clicking the **Refresh Now** button updates the execution plan instantly.

**Procedure**

1. In the **Execution Plan** window, select the desired node in the left pane.
2. Click the **Execute and Show Statistics** button.

**Results**

When you click the Execute and Show Statistics button, the display of nodes changes. In particular, each name is followed by some text that looks like (N) or (N, M%).

The N is the actual number of rows produced by that node. The M is the percentage of elapsed time that the node used to process the data. If M is not shown, it means that M is 0. For example, if the elapsed time was 60 seconds and if M is 20, it means that the node accounted for 12 seconds of the elapsed time (that is, 0.20 x 60). If M is 0, it does not necessarily mean that the node took no time at all. It means that the node did not contribute any delay to the elapsed time. This can happen if the node was processed by a background thread and processing was completed before the rows were needed by the parent node. Essentially, the M percentages help you determine which nodes to focus on if you want to make your query go faster. If you can reduce the M percentage at a node by some amount, the query could run faster by that amount.

---

**Cost-based query optimizer**

The IBM Cognos Virtual View Manager SQL Query Engine may utilize statistics gathered from the data sources, when they are available, to create an efficient SQL query execution plan for joins or unions across tables.
Statistics gathered on tables or cached views provide the SQL Query Engine with estimates on the table cardinality (the number of unique values or rows) so that SQL JOIN optimizations may be applied.

Cardinality statistics may be gathered on cached views and tables.

One of four statistical gathering modes may be enabled for the entire data source or only for selected tables and columns of data.

- **Do not gather statistics**—the default setting for all data sources and tables.
- **Gather table boundary statistics**—counts the number of rows in tables. It performs a SELECT COUNT (*) for the specified tables.
- **Gather column boundary statistics**—gets table boundary statistics and gathers the minimum, maximum, and counts distinct values for numeric data types. For string data types, the number of distinct values is calculated for tables with less than ten thousand rows.
- **Gather all statistics**—all the above plus a full table scan of numeric data types to build a histogram and for string data types a string index is built.

When statistics are enabled, the data source icon in Virtual View Manager provides a visual indication that statistics are enabled for use.

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 may be exported/imported when the data source is exported or imported or just the configurations may be backed-up.

Any user with the rights to Access Tools and Modify All Resources 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 the tables in the data source.

**About creating statistics**

To enable statistics gathering on one or more tables and for one or more columns in those tables, you create statistics on the parent data source and later save that data source after configuration.

You create statistics from the **Cardinality Statistics** tab which is available for any data source in Virtual View Manager.

**Note:** The data source login/password must be saved to enable statistics gathering.

**Cardinality statistics tab**

This tab contains the options that control how statistics are created.

Check the **Enable** box to allow use of the table statistics that will be gathered for the generation of optimized execution plans. The **Enable** check box need not be checked to configure statistics gathering, but use of the statistics is prevented when it is not checked.

Disable statistics gathering and use of any statistics that are already present from the data source at any time by clearing the **Enable** check box and saving.
Statistical data and all other data source statistics configurations may be deleted by clicking the **Destroy Stats** button ( ), in the upper right-hand corner of the data source **Cardinality Stats** tab page. **Destroy Stats** also deletes all table and column statistical data and associated configurations.

Data source tables inherit the data source statistics gathering mode after the first save of the parent data source. After the first save each table and individual column may be set to gather no statistics, use the data source settings, gather only table boundary statistics or to independently specify gathering for specific columns. By default, the data source settings are used for all tables within that source.

The System data tables (/services/databases/system/SYS_STATISTICS) may be consulted to see the status and the metadata (if you have the right to Read All Resources) about the successes and/or failures of statistics gathering from both the data source and individual tables.

**Data source statistics gathering**

This section describes how data source statistics are gathered.

**Modes**

As stated previously, one of four statistical gathering modes may be enabled for the data source

- **Do not gather statistics**—the default setting for all data sources and tables.
- **Gather table boundary statistics**—counts the number of rows in tables. It performs a SELECT COUNT ( ) for the specified tables.
- **Gather column boundary statistics**—gets table boundary statistics and gathers the minimum, maximum, and counts distinct values for numeric data types. For string data types, the number of distinct values is calculated for tables with less than ten thousand rows.
- **Gather all statistics**—all the above plus a full table scan of numeric data types to build a histogram and for string data types a string index is built.

Table statistics gathering settings for the statistical gathering mode will take priority over the more general data source setting. Initially all tables use data source mode settings.

**Number of threads**

**Number of Threads** specifies the number of threads executing queries that may be run on the database and Virtual View Manager. Each thread is run at low priority, but some databases will not tolerate lots of threads scanning tables at the same time. A full table scan for unique values can be a resource intensive process depending on the table size and the actual cardinality. Virtual View Manager will also have a thread capacity limit that may cause disk paging when nearing memory thresholds.

**Schedule**

Statistics gathering schedules may be manual, performed one time, or performed on a periodic basis. The scheduler is intuitive and easy to use, but here are a few notes on usage.
• **Manual**—Click the **Gather Now** button to initiate collection of statistics right away.

• **Exactly Once**—Requires that you set a day and time in the future to run.

• **Periodic**—Statistical collection will occur at timed intervals. The setting that probably makes sense for many data sources will be a weekly interval at an off-peak hour sometime past midnight. The setting that will make the least sense for a full table scan will be a very short interval every few seconds.

  Frequency of statistics gathering will be dependent on the volatility of the data in a table. Some tables never change and others change frequently. Gather statistics based upon the expected frequency of cardinality and table boundary change.

  **Note**: Utilize off-peak hours for gathering data source statistics as column boundary analyses and retrieval of all statistics requires significant resources to build histograms and string indices of table cardinality from all data present. Multiple threads may be engaged for data sources supporting resource intensive parallel processing.

• **Stats Gathering Timeout**—Specifies 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 is reached after table cardinality statistics processing is already returning data from a full table scan, then this timeout setting will trigger an attempt to gain some useful statistical estimates by making an assumption that the subset of data already returned is representative of the whole table even when only a partial result set is returned.

### Data source > Table > Cardinality Statistics tab

This section describes the different elements on this tab.

**Status Refresh Rate** checks the status table to show the last load time and current status. Any error in refresh of statistical gathering will be displayed as well.

**Manual Cardinality Overwrite** allows for manual specification of cardinality estimates if a table and the SQL conditions applied on selects are limited and mostly static. Entering integers here will invalidate statistics gathering for the table and the SQL Query Optimizer will use these numbers for generation of optimized execution plans. When this section is used the **Expected Cardinality** value is required, whereas **Minimum Cardinality** and **Maximum Cardinality** are optional.

- **Minimum Cardinality**—the minimum number of rows you would expect to get in return for a SQL SELECT on this table. Zero (0) is a valid number.
- **Expected Cardinality**—typical number of unique rows returned from this table
- **Maximum Cardinality**—maximum number of rows the resource could return

**Table Statistics Gathering** settings may be set to one of the following:

- **None**—does not gather statistics for the table
- **Use datasource settings**—datasource settings will show an action on columns (on save) only if the made performs actions on the columns.
- **Gather table boundary statistics**—counts rows in the table
- **Specify gathering for specific columns**—after this selection is made, use the Column Name | Gathering Rule table to set the Gathering Rule for individual table columns. Infrequently used columns may be set to **Do not gather statistics** when they rarely become part of the condition for select statements. Gather all statistics to build a numeric type histogram or to build a full string index for
string data types. BLOB, CLOB and a few other data types that would yield meaningless statistical data will not be configurable for statistical gathering.

Set All buttons

These buttons enable quick setting of the Gathering Rule for all columns of the table when the radio button is selected. After use of any of the buttons make further modification to individual columns as required.

Any changes must be saved to apply to the next statistics gathering mediated by the schedule set at the data source level.

Stats gathering time out

When the table statistics gathering time out is set to "-1" the data source statistics gathering timeout is used. A value of 0 (zero; 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 will not stop the processing immediately as the timeout will trigger an attempt to gain some useful statistical estimates by making an assumption that the subset of data is representative of the whole table even when only a partial result set is returned.

Cached execution plans

Pre-existing execution plans will prevent generation of optimized execution plans that are based on new statistics gathered on data sources and tables. Execution plans are generated the first time a SQL statement is executed and also any time a single character in the query changes.

Force a refresh of an individual query execution plan by adding a space to the SQL of a View or Stored Procedure. Flush out the entire query plan cache for everything by temporarily toggling the Query Plan Cache Enabled key in the Configuration window.

Built-in procedures

Three built-in procedures are available for using resource statistics.

- localhost/lib/resource/RefreshResourceStatistics
- localhost/lib/resource/CancelResourceStatistics

To get the description of any of these procedures, open it by double-clicking, and click the Info tab on the editor that opens on the right.

Temporarily changing the query plan cache setting

The Virtual View Manager server caches view execution plans. If View SQL utilizes data sources with fresh cardinality statistics, then the query plan cache may need to be purged or refreshed.

Purge the query plan cache by temporarily changing the Query Plan Cache configuration setting. Changing the query plan cache setting will force a one time regeneration of all execution plans.

Procedure

1. Select the Administration > Configuration menu option.
2. In the **Configuration** window, select **Virtual View Manager Server > SQL Engine > Caches > Query Plan Cache Enabled**.

3. In the right pane, select **True** or **False** to set the caching setting, and click **Apply**.

4. Click **OK**.

   Refresh an individual query plan cache by changing the View SQL superficially. Changing a single character in the SQL of a View or Procedure will force creation of a new query execution plan.

**How to change the View SQL superficially**

Suppose that the following is the SQL that has already been executed and its execution plan has been cached.

```
SELECT * FROM ds_orders
```

You can modify the SQL slightly by introducing a space or any character between the SELECT keyword and the identifier as follows:

```
SELECT * FROM ds_orders
```

Save the change and next execution will utilize any cardinality statistics available to optimize query plan execution.

---

**Defining and publishing indexes**

Indexes help to speed things up when searching or retrieving data from a database table.

Typically, when an index is created for a column, all the data in that column is scanned and a data structure called the index is constructed; the index makes the data in that column faster to look up, though indexing also has the potential to slow transactions that insert rows into indexed tables. The contents of the index are automatically updated whenever the data in a row are changed or modified.

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 the system ensures that every value in that column is unique. It will ensure this uniqueness for every update and insert.

In **Virtual View Manager**, the Indexes panel in the view editor enables creation of metadata labels for existing indexes and primary keys already present in the data source. Mark a column in a view as either indexed or as a primary key. Identifying indices and primary keys enable the query engine to employ logical algorithms for potentially faster more efficient joins that leverage organization of the datasource.

Virtual View Manager does not build the specified data structure, but creates a metadata definition that allows for more logical use of the existing resource data structure. Queries and joins don't necessarily run faster either 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 indeed has unique values.
Clients can benefit from knowledge of indexes and primary keys feature as this information is made available when the data service is published. More sophisticated client programs can generate efficient queries based upon column indices and primary keys.

Unfortunately, in some cases, it is not possible for the server to 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 have to go through a Java custom procedure and there is no way the server could analyze the procedure code and determine that the return value will continue to be unique and therefore still be a primary key.

**Procedure**

1. Open a view, and click the **Indexes** tab in the view editor.
2. Click **Add**.
3. In the **New Index** window, enter a name for the index, and click **OK**.
   The index is listed along with the column projections in the view.
4. Select the desired 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 notice the synchronization between the list of available columns and indexed columns.
5. Select the **Unique** box or the **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) Click **Add** to mark more columns as indexes, and click **Remove** to delete any columns that may have been improperly marked as an existing index.
7. Save the view.
8. Publish the indexed view in order to make the index available for external clients.
9. Open the system table **ALL_INDEXES** (in **Virtual View Manager Data Services > Databases > system**).
10. Click **Show Contents** in the Info tab to view the details of all the available indexes in the system.
11. If you have published data source tables, the indexes on those resources would also be listed in **ALL_INDEXES**.
12. Click a row in the Result panel output to view the result details.

**Defining foreign keys**

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 displays in Table A.

Foreign key relationships indicate that if you join A.X = B.Y, there will be exactly one row in Table B found for each row in Table A. This information provides a useful hint for people and tools performing join operations.
In IBM Cognos Virtual View Manager, the Foreign Keys panel in the view editor lets you create a definition that acknowledges and allows use of any foreign keys that already exist in the data source.

**Procedure**

1. Open an appropriate view.
   - The view illustrated 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 view editor.

3. Click **Add**.

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**.
   - Notice that the columns that are available for you to mark up as a foreign key column are now listed in the section **Available Columns**. These columns are exactly the same columns selected for projection in the view. 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.

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.
   - Notice that the columns in the **Parent Table** are now 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 view.
   - 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 in this view.
   - 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.

9. Publish the view.
   - For details on publishing, see [Chapter 8, “Resource publishing,” on page 147](#).
   - Now, you can also verify that the foreign keys you defined are accessible to external client applications.

10. Open the system table /services/databases/system/ALL_FOREIGN_KEYS, and view its contents.
11. Right-click **Virtual View Manager Data Services/Databases/system/ALL_FOREIGN_KEYS**, and click **Show Contents**
    - The foreign keys you defined are listed in the **Result** panel.
Chapter 11. Caching resources

Caching locally stores results returned from views and procedures to improve performance response times and to protect data sources from overuse. Caching is not meant for simply replicating or storing source data.

When should caching be used? Caching is best used for materialized views when:
- A view or procedure requires substantial execution time
- The data does not change significantly within a given period
- The data source should be protected from excessive usage

There are two types of cache storage: file and database. File caching allows for quick configuration and usage of the entire materialized view providing dramatic cuts in response time when used properly, but with significant potential for slowed responses if it is misused. Database caching enables result set storage in a database so that further manipulations of the result set may be performed while dramatically cutting response times.

Caching stores the results from views or procedures that meet the above criteria. However, if any of the results of those same views and procedures are to be used by other views, or be filtered or sorted, then the cache should be stored in a database.

The following tabular resources can be cached:
- SQL views
- Physical tables that have been introspected

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 following resources cannot be cached without being wrapped in one of the resources itemized above:
- 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, definition sets, and so on

View caching

Caching a view is a straightforward process that saves the data results from the execution of a view.
**Procedure caching**

Procedures may be cached just like views, but because of the variety of inputs and outputs the implementation is slightly more complex.

The caching mechanism for a Procedure varies depending upon whether the Procedure has input and/or output parameters.

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, and one for the second cursor.

If a procedure has input parameters, the cached results are tracked separately for each unique set of input values. For example, a procedure with an INTEGER input parameter would have separate results cached for inputs of 1, 3, and 5. A procedure with two INTEGER input parameters would have separate results cached for inputs (1,1), (1,2), and (2,5). Each unique set of input parameter values is called a variant.

**Note:** A procedure cache that utilizes non-null input parameters must be seeded with at least one variant from a client application for the Cache Status to change from NOT LOADED to UP. Using the Refresh Now button will not change the status of an unloaded cache. Procedure caching configuration may be correct, but the status will not show that it is loaded until a client seeds the cache.

When a procedure cache is either manually refreshed or when it is refreshed using the RefreshResourceCache procedure all the cached variants already in the table will be refreshed. If no variants have yet to be cached, then either nothing is refreshed or only the null input variant is refreshed.

The maximum number of variants to cache at one time is configured on the Cache tab. The default value is 32. When the system exceeds this number of variants, the loading of a new variant will cause the least recently used variant in the cache to be discarded.

**Cache storage type**

The data returned from a view or a procedure can be stored automatically as a file or in a data source table specified by the user.

The automatic storage option (file-based storage) can improve performance and protect data sources from overuse, but only if the results are not sorted or filtered for use in other views. If a file cache is used in the wrong circumstances it could significantly degrade performance.

It is much preferred if a cached resource is stored in tabular form in a data source. The schema of the resource data must exactly match the schema of the selected data source table. There can be more columns in the database than in the resource data but the data types of the columns in both must match. Luckily the DDL may be generated to create a caching table that exactly matches your view or procedure output.
File cache limitations

File caches do not store index information, nor do they allow for SQL logical operators and filters to be pushed to the data source. Since 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. Any SQL operations are executed on the cached data within Virtual View Manager memory.

**Note:** You should not build additional queries on top of large file cache, as this can use a significant amount of memory.

For examples, suppose a view, V, represents a million rows and your query is

```
SELECT name FROM V WHERE id=1
```

If V is not cached, the "id=1" filter would be pushed to the data source and Virtual View Manager only needs to process a single row from the data source. If V is cached, Virtual View Manager will be forced to read a million rows of data from the cache (if there are 100 columns, all 100 columns are read into memory as well) and evaluate the filter condition on every single row to produce that single row.

Setting up caches

For the resource being cached (view or procedure), storage will depend on result set output (scalar or cursor) and on usage (retrieval of the entire result set or retrieval of a filtered subset).

See the previous topic for a discussion of potential drawbacks. Decide upon a storage type (system-created file or data source) for caching the execution result.

**Procedure**

1. In Virtual View Manager, open the view or procedure.
2. Click the **Caching** tab.
3. Click **Create Cache**.
4. Under **Status**, select the **Enable** check box.
   - When a cache is disabled, all the existing cache settings are ignored. The view or procedure is used as if caching did not exist.
   - When a cache is enabled, use of a view will make use of the cache storage table to find the data, and use of a procedure will make use of one or more cache storage tables to find the data.
   - Toggling between the enabled and disabled state will not cause refreshing or clearing of the data. It will also not reset the expiration date for the data.
5. Under **Storage**, specify the storage type as either **Automatic** or **User Specified**.
   - **Automatic**: The cached data will be stored in a system created file data source at `<localhost>/lib/sources/cacheDataSource`
     - Each cached resource has a cache table in the cacheDataSource data source. The cache table's name contains the resource type (view or proc for procedure) and a system-generated random ID. The **Annotation** section of the **Info** tab contains the full path to the cached resource.
     - Go to step 7.
   - **User Specified**: The cached data will be stored in a specified data source. Specify such a storage data source for the cache storage:
Use the **Browse** button adjacent to the **Data Source** field to locate and specify the data source. Once you select the data source, its full path will be displayed in the **Data Source** field.

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. In the **Caching** section, select the **Browse** button for **Status Table**. In the window that opens, select the data source, supply a name for the status table in the **Create Table** field, and press **Create**. The DDL for creating the table will be displayed, and you can press **Execute** to create the table. Repeat this procedure for creating a tracking table using the **Browse** button for the **Tracking Table** field.

Not all data sources that can be used for cache storage might support the generation or execution of DDL statements. In some cases, DDL may be generated and presented but use of an external tool may be required. In other cases, even the display of DDL may not be available.

- Navigate back to the **Caching** panel.
- Use the **Browse** button in the **Table for Caching** section to specify a location within the data source (that you just selected as your storage) to create a table for caching. This location can be the root level of the data source or a schema.
- Save the cache.

6. Under **Refresh Mode**, specify the cache refresh mode. Refreshing a cached resource retrieves data from the source(s) and clears stale data as specified.

If a resource uses a data source that was originally added to Virtual View Manager using pass-through mode without saving the password, row-based security may affect the cache refresh functionality. For example, suppose that a cached view, named CachedCommonView, uses the SQL `SELECT * FROM db2.T1;` and the user John with the password john1 is allowed only to view 10 rows, whereas the user Jane with the password jane1 is allowed to scan 20 rows from table T1 in the data source db2. Every time Jane refreshes the view cache, both Jane and John would be able to view 20 rows, but every time John refreshes the view Jane can only view 10 rows.

- To refresh the cache immediately, click **Refresh Now**.
- To refresh the cache manually, select **Manual**.

  When the caching refresh mode is set to **Manual**, the resource owner or an administrator must manually refresh the cache.

- To refresh the cache just once at a specific time, select **Exactly Once**, and specify the time to start caching in the set of drop-down boxes in the section labelled **Start on**. The **Hour** field will accept an entry of 0 or 12 for the midnight or the noon hour depending on the value of AM/PM.

- To refresh the cache periodically, select **Periodic** and specify in the **Refresh every** section, how often to execute the resource for caching: every number of seconds, minutes, hours, days, or weeks. In the **Start on** fields, specify the time and date respectively to execute the resource.

  The date entered indicates the time at which the first occurrence of the caching event will 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.

  Each time a refresh completes, it starts a timer and performs the refresh when the timer completes. This timer persists across server restarts. If the server is down when the timer should have completed, the refresh will be performed shortly after the server starts.
A start time can be specified if the refreshing should not begin before a given
time in the future. This defaults to the current date and time.

The start time can be used to place the recurring refresh at a given time of
the day. For example, if the start time is 2am tonight and the interval is 1
day, it will refresh every day at around 2am. The actual time will drift by the
time it takes to start the background refresh task (usually milliseconds, but
on a loaded system this can be longer).

If the refresh interval is shorter than the duration of a refresh, a scheduled
refresh attempts while a refresh is still running will be ignored.

Any time the start time or interval is altered, the schedule is reset. If the start
time is in the past, this can cause an immediate refresh. For example,
changing the interval from 4 hours to 2 hours will cause an immediate
refresh if the start time is in the past. Disabling and then re-enabling the

cache will also reset the schedule with the same behavior.

7. Under **Expiration Schedule**, specify when the cache will expire.
   Select **Never expire** or **Expire after** and a specified number of seconds,
   minutes, hours, days, or weeks. The expiration period applies from the end of a
   successful refresh.

   Any resource can have an expiration period applied to it. For example, 1 hour.
   After this period, the data is considered expired and will not be used. Any new
   query attempting to access the data will discard the old data and trigger a
   refresh.

   For tables, the expiration period applies to the whole table. For procedures,
   each input variant's data has its expiration tracked separately.

   In addition to having the data discarded if an attempt is made to access it, a
   background task will look for expired data and proactively delete it. This
   background task will check every 60 seconds or the shortest expiration period
   assigned to any enabled cache, whichever is larger.

   For example, if the shortest period is 1 hour, then it will run once an hour. If
   the shortest period is 10 seconds, then it will run once every 60 seconds. If no
   enabled cache has an expiration period, then this task will not run.

8. Under **Advanced**, specify when to clear the cache and the maximum number of
   procedure variants:
   - **When user clears it manually**—clear the cache only when cleared explicitly
     through the Clear Now button, an API call, or on cache expiration.
   - **When refresh fails**—clears the cache if a refresh fails. The effect of this
     option is to allow access to previously cached data during a refresh, but to
     either provide updated data or to end up with a cleared cache when the
     refresh completes with success or failure respectively.
   - **When refresh begins**—automatically clears the cache before starting a
     refresh. The effect of this option is that any client attempting to read from the
     cached data will not see the previously cached data and will wait for the
     new data.
   - **Maximum number of procedure variants**—(for procedures only) maximum
     number of unique set of input parameter values.

9. Save the cache settings.

**Cache indexing**

The Virtual View Manager server does not index cache tables automatically. The
user must manually specify indexes according to the data result sets and usage
patterns so that cache performance may be improved.
Indexing specific columns in the result set of a cached view will yield performance benefits when those columns are commonly used for filtering and viewing some subset of the data from the cache.

Create an index for the cachekey column as a specific cache key filter condition is appended to the SQL sent to the cache table every time a cached result is requested. The cache key is a unique number generated for each cached result set; it is the only caching table column automatically generated by the server. Because the cache key filter condition is a highly specific query predicate it will always be advantageous if the cachekey column is the first column of the index.

Indexing the cachekey column is most useful in cases where many procedure cache variants are present and commonly used queries do not return a large subset of rows.

Adding an index does add overhead to cache refreshes, but may benefit cache result retrieval performance, depending on the individual query parameters. Performance testing ensures that performance goals are reached when specifying cache indexes.

### Cache configuring and enabling

When a View or Procedure is selected for being cached, the cache needs to be configured and enabled. This section describes how to perform these tasks using the Virtual View Manager interface.

For details on how to do these tasks programatically, see the section “APIs” on page 207.

To configure caching for one of the supported resources, open the resource in Virtual View Manager and select the Caching tab in the resource editor. You will see a button labeled Create Cache. Pressing this button will configure the resource for caching (as of the next save operation), but leave the caching in a disabled state. The enabled/disabled state can be toggled using the Enable check box on the Caching panel. When configured and enabled, the cache will be activated. When disabled, the cache configuration and any data in the cache will be preserved but any use of the resource will execute as normal instead of using the cached data.

To de-configure caching, press the Destroy Cache button on the upper right corner of the Caching panel. This action will erase the configuration and clear the cached data. There is an important distinction between a cache being configured or de-configured and enabled or disabled.

- Any cache that is configured will be tracked on the Cached Resources panel in the Manager and have data that is cached, preserved, and tracked. A cache that is de-configured is not tracked any longer and the data in the cache is cleared.
- This is distinct from being enabled or disabled, which is a simple switch on whether or not to use the cache right now.

### Behavior in Virtual View Manager

Executing a View in Virtual View Manager will read from the cache if the View is saved and caching is enabled. When showing the execution plan, the cache data table will be shown in the plan.

For more information about cache data tables, see “Cache data tables” on page 209.
If the View is edited and not saved, the View will be run anonymously and the caching behaviors of the View will be ignored.

Executing a Procedure will read from the cache for all Procedures except SQL Scripts. Virtual View Manager always runs SQL Script procedures in an anonymous way that bypasses the caching behaviors. To test the caching behavior of a SQL Script, it must be saved and must be called from another Procedure or View.

**Automatic and data source caching**

This topic discusses how cached data are stored.

**Automatic caching**

When caching is first configured, it defaults to automatic caching.

In the **Automatic** mode, the storage data source and storage tables for the cached data are chosen automatically for you.

The storage tables are automatically created, dropped, and maintained as the signature or existence of your Views and Procedures change.

Automatic caching makes use of a file-based data source located at: /lib/sources/cacheDataSource. The actual data is stored in files under installation_location/tmp/cache. For details on file-based data sources, see the section [“The ‘File-Cache’ data source” on page 201](#).

The cache storage table names are automatically generated to be unique using names like view12345 and proc54321p2. The table names chosen by the system are displayed on the **Caching** panel for the resource. Also, the annotation on each of the generated tables includes information on which View or Procedure is being cached to that table.

These tables are accessible directly, but in general, you should not use these tables directly. Having them exposed makes it possible, however, to view the contents of these tables when developing.

When the server is running in a cluster and the cache is in **Automatic** mode, each server will keep a separate copy of cached data on its local file system. No sharing of data is performed.

**Data source caching**

As an alternative to the automatic mode of caching, cache data can be stored in a relational database.

To use this kind of storage, select the **User Specified** option in the **Caching** panel and either type the path of the data source or use the Browse button to select one.

When the server is running in a cluster and the cache is in **Data Source** mode, all servers in the cluster will access the same data source for cache information. In addition, the servers will 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 will perform the refresh, then both servers will use the updated data.
Status and tracking tables
Before a data source can be used for caching, it must have two types of tables set on it. These are the Status tables and Tracking tables. Only one pair of tables is required for the data source to support any number of Views and Procedures being cached to the data source.

If the table you need does not already exist, you can create them. Open the data source in Virtual View Manager and select the Info tab. The Caching box in the Info panel will show two input fields, where the table paths can be entered, and a Browse button for each input field. Use the Browse button and select a container, which can be the data source node itself or a folder in the data source, type in a table name, and press the Create button. This command will display the DDL to create the table. This DDL can be copied out and run in a database-specific tool or the Execute button may be pressed.

To configure existing storage tables, open the data source in Virtual View Manager and select the Info tab. In the Caching box, type the table paths, or use the Browse button to choose a table.

The colors red and black indicate valid choices and tables can be created or recreated as supported by the data source.

The paths to the tables are case sensitive. Be aware that when creating a table in DDL the database may not preserve the case of your entry and so when the data source and text entry are updated after running the DDL, the case of the created table name may not match what was originally entered.

Storage tables
Once a data source is chosen for storing cached-data, each of the required storage tables needs to be specified.

In the Caching panel for a resource, one field will be present for each result cursor. For a Table or View one table is needed, and this table name is displayed in the field labeled result. Procedures may need more than one output table.

Either type the path of the table you want to use, or use the Browse button to select one.

When using the Browse functionality, tables are highlighted in red if they are not compatible with the data you need to cache. Selecting one will display a note detailing why the table is not compatible. Tables shown in black are compatible and can be used successfully.

If you select an existing table that is not compatible, a button to Recreate the table will also be shown. Selecting this button will display the database specific DDL that needs to be executed to recreate the table with a compatible schema for storage. This DDL can be copied out and run in a database specific tool, or the Execute button can be pressed to attempt and execute the DDL using the data source's configured login information. This attempt may or may not succeed depending on the permissions the data source's login has within the database. Be aware that “recreate” is accomplished by DROP and then CREATE actions so any data in the existing table is lost.

If the table you need does not already exist, you can create it as described in the preceding section.
In either case, the execution of DDL will be followed automatically by the equivalent of an Add/Remove Resources operation. The newly created or recreated table will be updated, but other resources in the same folder in the data source may also be updated.

The paths to the data source and table are case sensitive. Be aware that when creating a table in DDL the database may not preserve the case of your entry and so when the data source and text entry are updated after running the DDL, the case of the created table name may not match what was originally entered.

### Privileges

In order for someone to use the View or Procedure that is being cached to a data source, that user must have proper privileges granted to access both the Status Table and data tables.

Users that need to read from the cache should be granted the SELECT privilege on these tables and READ privilege on folders above these tables.

The user that owns the resource being cached is the user identity that cache refresh and clear operations will be run with. That user needs the READ, SELECT, INSERT, UPDATE, and DELETE privileges on the Status Table, Tracking Table, and the data tables.

Privileges are managed automatically when using the Automatic mode of caching.

### The "File-Cache" data source

The automatic mode of caching makes use of a pre-created instance of the File-Cache data source type that cannot be re-configured.

Additional instances of this data source can be created using the New > Data Source menu option just like for any other data source. These additional instances can have the storage directory set up to store cache data files anywhere on the file system. These additional instances will not be managed the way the Automatic caches are. All table management is handled exactly like it is for any other data source.

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 for best performance with the server.

The directory tree of files is specifically designed to be free of server-instance specific data. This means it is possible to copy this directory to a new machine, re-introspect the data source on that machine, and have the data display. This allows for the design of portable cached data sets.

You should not use the File-Cache data source or other file-based data sources when in a cluster. The caching system assumes that data source caches are available to all servers and this is not true for file caches. You should also not use network-mounted files because the file data sources do not support file locking.

### Cache data management

This section discusses managing cache data.
**Terminology**

The term refreshing or loading a cache means to read all the data from the source table or procedure and create a new set of data in the storage table(s). This data set replaces the previous data set if the refresh succeeds.

The term refresh is not meant to imply any sort of incremental updating behavior. Incremental updates are discussed in the topic ["Incremental caching” on page 212](#).

The term clearing a cache means to remove all data from the storage table(s) so there is no longer any data in the cache.

**Scheduled refresh of cached data**

You can schedule a refresh of the data. This schedule can be set to any number of seconds, minutes, hours, days, or weeks.

You schedule a refresh of the data in the Caching panel. You can also refresh the cached-data programmatically. See the topic ["APIs” on page 207](#).

Each time a refresh completes, it starts a timer and performs the refresh when the timer completes. This timer persists across server restarts. If the server is down when the timer should have completed, the refresh will be performed shortly after the server starts.

A start time can be specified if the refreshing should not begin before a given time in the future. This defaults to the current date and time.

The start time can be used to place the recurring refresh at a given time of the day. For example, if the start time is 2am tonight and the interval is 1 day, it will refresh every day at around 2am.

If the refresh interval is shorter than the duration of a refresh, any scheduled refresh attempts while a refresh is still running will be ignored.

Any time the start time or interval is altered using Virtual View Manager or any API, the schedule is reset. If the start time is in the past, this can cause an immediate refresh. For example, changing the interval from 4 hours to 2 hours will cause an immediate refresh if the start time is in the past. Disabling and then re-enabling the cache will also reset the schedule with the same behavior.

**Expiration of cached data**

Any resource can have an expiration period applied to it, for example, "1 hour". After this specified period, the data is considered "expired" and will not be used. Any new query attempting to access the data will discard the old data and trigger a refresh.

The expiration period applies from the end of a successful refresh.

For tables, the expiration period applies to the whole table. For procedures, each input variant's data has its expiration tracked separately.

In addition to having the data discarded if an attempt is made to access it, a background task will look for expired data and proactively delete it. This background task will check every 60 seconds or the shortest expiration period assigned to any enabled cache, whichever is larger. For example, if the shortest
period is "1 hour", then it will run once an hour. If the shortest period is "10 seconds", then it will run once every 60 seconds. If no enabled cache has an expiration period, then this task will not run.

**Rules for clearing caches**

Virtual View Manager offers three options for clearing a cache.

- **when user clears it manually** - This option means that the cache clears only when cleared explicitly through Virtual View Manager or an API, or when the cache expires.
- **when refresh begins** - This option automatically clears the cache before starting a refresh. The effect of this is that any client attempting to read from the cached data will not see the previously cached data and will wait for the new data.
- **when refresh fails** - This option automatically clears the cache if a refresh fails. The effect of this option is to allow access to previously cached data during a refresh, but to either provide updated data or to end up with a cleared cache when the refresh completes with success or failure respectively.

**Triggers**

This section describes using triggers with caches.

**Triggers that operate on caches**

Trigger resources can be used to cause caches to refresh on a scheduled basis.

You can use triggers to create richer refresh schedules than the built-in scheduling feature on the Caching panel for resources.

Triggers can also be used to cause refreshes to occur in other circumstances. For example, a user-defined event may be used to cause a refresh.

**Triggers that detect cache changes**

There are two System Events related to caching that Triggers can be made to listen for: cacheRefreshSuccess and cacheRefreshFail. A Trigger can run logic as a result of either of these events.

**Transaction result caching**

There is a related feature in the product called "transactional result caching" that is available for Procedures. This feature is enabled by checking a box on the Info panel of any procedure.

When enabled, the first time the procedure is run during any transaction will capture the results in memory (using a disk-backed temporary store if enough memory is not available), and additional calls to the procedure with the same input parameters will return the same data.

This feature is not the same type of caching that is used throughout this document. No refresh, clear, status, or tracking features are available.

The Transaction Result Caching feature may 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.
**Cache lifecycle**

This topic describes the details of the cache lifecycle. It is intended for those that want to understand the timing and states in close detail.

Throughout this description, the term cache should be treated as applying to the whole cache for a View or to the cache data for a single input variant for a Procedure unless otherwise specified.

**Enabling and disabling**

When a cache is disabled, all cache settings are ignored. The View or Procedure is used as if caching did not exist.

When a cache is enabled, use of a View will instead make use of the cache storage table to find the data, and use of a Procedure will make use of one or more cache storage tables to find the data.

Toggling between the enabled and disabled states will not cause refreshing or clearing of the data. It will also not reset the expiration date for the data.

**Loading and refreshing**

A cache can be in a not-loaded state or loaded state.

**Not loaded state**

Initially, a cache starts in a not-loaded state.

**Forced loading of a cache**

A cache can be forced to load in four ways.

- An explicit Refresh Now action is requested
- A scheduled refresh
- Calling the system built-in procedure /lib/resource/RefreshResourceCache()
- Executing the published RefreshResourceCache Web services operation

All of the above are non-blocking actions that cause a background task to begin a refresh action if a refresh is not already in progress. If a refresh is already in progress, the action will simply return immediately.

For procedures, a forced refresh will refresh all variants that are currently cached. If no variants are cached, then no action is taken.

**On demand loading of a cache**

In addition, a refresh may be caused indirectly if any attempt to read from a cache that is not loaded.

When this happens, it will start a background task to refresh the cache if a refresh task is not already running, then block until the refresh completes. If the refresh completes successfully, the query will unblock and use the data. If the refresh fails, the blocked query will fail providing the refresh failure's information as the reason for the failure.

For procedures, only the one variant being accessed will be refreshed. This is different from a forced refresh, which refreshes all variants.
Reading from a loaded cache
Any attempt to read from a cache that is already loaded will return the current data, even if a refresh is currently in progress.

The refresh task
The background task that performs the refresh always runs as the user that owns the View or Procedure that is being cached. This is true regardless of which user requests or causes the refresh task to start.

Interaction with pass through data sources
The cache system does not support pass through data source configurations. The refresh action is always performed as a single user and is always performed without that user's credentials. Because of this, all data sources underlying the View or Procedure being cached must have specific credentials coded on them or the refresh action will fail.

If pass through is enabled on a data source that is under a cached view, the cache will no longer be able to refresh and whatever data is in the cache will be used unless the clear on refresh failure clear rule is also used to force a clear.

Cache clearing
This section discusses the ways to mark a cache for clearing.

A cache can be manually cleared in three ways:
- An explicit Clear Now action is requested.
- Calling the system built-in procedure /lib/resource/ClearResourceCache()
- Executing the published ClearResourceCache Web services operation

Cached rows are marked for clearing as a background task. A new request for that data will detect data marked as stale and fetch fresh data from the source(s). A forced clear marks all cached variants as stale data and readies them for clearing. It does not cancel any in-progress refresh actions. Those refresh actions will continue to execute and when successful they will load the cache as expected.

Other dynamic causes for a cache to clear
A cache may also be cleared in the following cases.
- An attempt to read from the cache while the cache contains expired data.
  The test for expired data is performed prior to any read and causes any expired data to be immediately cleared. Since the cache is not loaded, it will cause a refresh as described in the Loading and Refreshing topic.
- An attempt to refresh the cache while it contains expired data. The test for expired data is performed prior to starting the refresh and the clear is performed before returning from the refresh action.
- An attempt to refresh a cache using the when refresh begins clear option is selected.
  When this clear rule is used, the cache is cleared prior to returning from the refresh action.
- An attempt is made to load a procedure variant that is not currently loaded and the maximum number of variants is already loaded.
  This causes the least recently used variant to be cleared immediately, then for the processing to continue as described in the Loading and Refreshing topic.
- A cache refresh fails and the when refresh fails clear option is selected.
When a refresh fails, instead of making new data available, the refresh task will clear out existing data.

In all the above situations, when it says that a cache is cleared immediately, it means that the data is made unavailable immediately by altering the Status Table and internal tracking information of the server, but a background task is used to actually clear the data from the data tables.

**The data clear task**
The background task that performs the data clear always runs as the user that owns the View or Procedure that is being cached. This is true regardless of which user requests or causes the clear task to start.

**Cache status**
The cache status is reported in two places.

For the resource, the cache status is reported on the Caching panel. In the Administrator, the cache status is reported on the Cached Resources console.

- A cache begins in the Not Loaded state. This means no data is loaded.
- A cache that has been loaded successfully is in the OK state.
- A cache that fails to load successfully is in either the Down or Stale state. The Down state indicates that the cache is not loaded and the most recent refresh failed. The Stale state indicates that the cache is loaded with valid data, but that the most recent refresh failed. When a cache is Stale, reads against the cache can succeed.
- A cache that cannot operate due to a configuration error is in the Config Error state.
- A cache that is disabled is in the Disabled state.

**Cache events**
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 reported events are:
- cacheEnable
- cacheDisable
- cacheClear
- cacheRefreshStart
- cacheRefreshEnd
- cacheRefreshFail

The logging levels for these events can be set in the Configuration panel as with all other supported events. Refer to the product documentation for additional details.

**Configuration changes**
Changing a cache’s configuration to point to a different storage data source or changing a storage data source to use a different Status table will cause the server to discard all in-memory information about the data being cached and for this information to be read from the Status table again.
This reread is performed as a background task, so there is a brief period prior to
this reread when accessing cache data may encounter errors.

Changing a cache's configuration to use different data storage tables will not cause
any in-memory information to be updated. The cache will use the new tables
assuming data is stored under the proper cache key. You should perform an
explicit Clear or Refresh action following this change.

Renaming a cached resource or any of its parent folders will cause the Status table
to be updated with the new name and path of the resource. This update is
performed as a background task, so there is a brief period prior to this update
when accessing cache data may encounter errors.

Changing a cached View or a Procedure's signature for a cache that is using
Automatic caching will cause the old data table to be dropped and recreated. An
explicit clear action is performed automatically. This recreate and clear is
performed as a background task, so there is a brief period prior to this task
completing when accessing cache data may encounter errors.

Changing a cached View or a Procedure's signature for a cache that is using Data
Source caching will not change the schema of the data storage tables. As a result,
the View or Procedure is likely to have a configuration error. The error state will
remain until the tables are either recreated using the Virtual View Manager
interface or are altered using external database tools and then re-introspected using
the Virtual View Manager interface.

Using the Virtual View Manager interface to execute DDL to recreate a table will
delete any data in the cache table. You should perform an explicit Clear or Refresh
action following this change.

APIs

The caching system can be operated on programmatically.

Procedures

The following procedures are available for use from SQL Scripts, Java Procedures,
or can be published for use from JDBC clients.

- /lib/resource/ClearResourceCache(path,type)
- /lib/resource/RefreshResourceCache(path,type)
- /lib/resource/UpdateResourceCacheEnabled(path,type,isEnabled)
- /lib/resource/CreateResourceCacheKey(path, type,cacheKeyOutput)
- /lib/resource/LoadResourceCacheStatus(path, type)

Web services

The following Web services are available for use from clients outside the server.
They are under the Virtual View Manager Data Services/Web Services/system/
admin service.

- resource/resourcePort/clearResourceCache
- resource/resourcePort/refreshResourceCache
- resource/resourcePort/getResourceCacheConfig
- resource/resourcePort/setResourceCacheConfig
Storage tables

This topic describes the tables for storing cached data.

The data source status table

The data source Status table is used to track which Views and Procedures currently have cached data stored, when they were last refreshed, and so on.

It is not legal in the current release for a single status table to be shared by two different Virtual View Manager Servers. Each server must have its own table.

Reference columns

The following columns are used to identify a particular set of cached data.

- **clusterid** - This column is NULL if the server it not a member of a cluster. It has the cluster’s ID if the server is a member of a cluster.
- **serverid** - This column identifies which server is responsible for the entry in the table. A server’s ID is constructed by appending its hostname, port, and a generated number. For example: myhost.mycompany.com-9400-12345
- **resourceid** - This column identifies which resource in the server is being cached. This is the path of the resource. For example, /shared/myfolder/myview.
- **parameters** - This column identifies which set of input parameters is being cached. This column is NULL for Views, and is always non-NULL for procedures. The value is a comma-separated list of the input parameters in a string form identical to the form that would be used to call the procedure from SQL Script. For example, if a procedure that has a VARCHAR and INTEGER input could have a parameters value of "a string value",42".

Status columns

The following columns are used to identify the status of a particular set of cached data.

- **status** – This column has the value ‘A’ if the row describes an Active (loaded) set of cache data. The value ‘I’ indicates an In Progress (refreshing) set of cache data and the value ‘P’ indicates a Probe by one server in a cluster against refresh attempts by other servers in the cluster. The value ‘F’ indicates a Failed set of cache data. The value ‘C’ indicates a Cleared set of cache data. The value ‘K’ is special and is described below.
- **cachekey** – This column holds the unique key that identifies a set of cache data. For an Active status row, this is the cachekey column value in the data table(s) for the active data. For an In Progress status row, this is the cachekey in use for refreshing.
- **starttime** – This column holds the time the data load (refresh) started.
- **finishtime** – This column is NULL while In Progress and is updated with the finish time when a load (refresh) ends in either success or failure.
- **cleartime** – This column is NULL at all times in the current release.
- **bytes** – This column is NULL while In Progress or after failure to load. When in an Active (loaded) state, this holds the approximate number of bytes in the cache.
- **message** – This column is NULL except for Failed rows, in which case this column holds the failure reason message.
The key row
There is one special row in the table with the status of 'K'. This row does not describe any cached data. Instead it is used to hold the next available cachekey value.

The server updates this row as it uses cachekey values. Currently it increments it by 1,000 each time.

The data source tracking table
The data source Tracking table is used to track the Views and Procedures that are currently using the data source for caching, and what tables in the data source are in use. This is purely informational and is not used by the server at this time.

It is not legal in the current release for a single tracking table to be shared by two different Virtual View Manager Servers. Each server must have its own table.

Reference columns
The following columns are used to identify a particular set of cached data.
- clusterid - This column is NULL if the server it not a member of a cluster. It has the cluster's ID if the server is a member of a cluster.
- serverid - This column identifies which server is responsible for the entry in the table. A server's ID is constructed by appending its hostname, port, and a generated number. For example: myhost.mycompany.com-9400-12345
- resourceid - This column identifies which resource in the server is being cached. This is the path of the resource. For example: /shared/myfolder/myview

Tracking columns
The following columns are used to identify the tables in use for of a particular set of cached data.

If a resource (such as a Procedure) requires more than one table for caching, one row will be entered in the Tracking table for each table it uses.
- catalog - This column identifies the catalog (if any) of a table in use.
- schema - This column identifies the schema (if any) of a table in use.
- table - This column identifies the table name (if any) of a table in use.
- createtime - This column is always NULL in the current release.

For data sources that do not support catalogs, the catalog column is always NULL. The same applies to the schema column.

Some database products refer to the schema concept as a database or by some other name used to segment tables into separate namespaces. For these database products, this name will be found in the schema column.

Cache data tables
Each cache data storage table contains one additional column in addition to the columns of data that are being stored. This cachekey column contains an integer that identifies which rows belong to which variant of the data.
The data source Status table identifies which cachekey value is associated with current data or with specific parameter input variants for procedures.

**Transaction isolation**

The cachekey value in the Status table 'A' row is known as the active cache key.

If a transaction is making use of a cache with a given cachekey value and a refresh occurs, the newly loaded data will get a new cachekey value and will be made active when the refresh completes. The rows associated with the key values still in use by one or more transactions will not be cleared until they are no longer in use. This allows existing transactions to complete with consistent data for their entire run.

A background task is used to clear cache data that is no longer in use.

---

**Best practices and use cases**

These topics discuss the best practices and use cases for using the caching feature.

**File versus database caching**

The main difference between file and database caching is that databases can perform result set filtering (WHERE clauses). This makes the file caching useful for small result sets, or for cases where the entire result set will be read all the time.

Because databases caching allows filter conditions (WHERE clauses) to be executed in the data source, larger result sets that are frequently filtered can perform much better than with file caching.

**Cache indexing**

The Virtual View Manager Server does not index cache tables automatically. It is up the user to specify indexes according to the data result sets and usage patterns so that cache performance may be improved.

**Note:** Adding an index does add overhead to cache refreshes as insertions may take slightly longer depending on the database used, but the performance benefits of faster result set retrieval from an indexed cache may make that overhead worthwhile. Do performance testing to ensure that goals are reached.

Indexing specific columns in the result set of a cached view will yield performance benefits when those columns are commonly used for filtering and viewing some subset of the data from the cache.

The first column of any index should be the cachekey column. The cachekey column is the only column automatically generated by Virtual View Manager. Indexing the cachekey column will be useful in cases where many procedure cache variants are present.

**How to avoid stale data**

Cache data can become stale (out of date) when the data in the original system changes.
Scheduled refreshing
One means of avoiding stale data is to schedule a refresh action to occur on a regular basis. Scheduled refreshes give control over when the cache data is loaded. It also results in the best performance for clients reading from the cache, because the refreshes occur as background tasks.

The down side of a scheduled refresh is that the cache is loaded regularly even if no clients are reading cache data.

Expiration based refreshing
Another means to avoid stale data is to give the data an expiration period. Expiration ensures that data is never more out of date than this period.

The down side of expiration is that cache refreshes may occur at any time and that the refresh blocks the client that detects that data is not loaded until the refresh completes. This can cause uneven response times for clients.

How to avoid unnecessary refreshing
If there is a way to find out if any data has changed in the original data source, a procedure can be scheduled to run on a regular basis that tests if data has changed.

If data has changed, it can call RefreshResourceCache() to force a refresh. If not, it can return without doing so. This has the advantages of a scheduled refresh, but avoids unnecessary work if no data has changed.

How to avoid load by caching
A common use of caching is to avoid putting undue load on a corporate data system.

By copying data into a cache, the cache absorbs the load. Both scheduled refreshes and expiration can be used to control how often the data is read from the original source.

Cache windows
When caching to avoid load, it is sometimes desirable to only operate the cache during specific hours. This can be accomplished by enabling and disabling the cache on a schedule.

To do this, create a Trigger resource that calls the UpdateResourceCacheEnabled() procedure on a schedule to disable the cache at the appropriate times. Then create a second Trigger resource that enables the cache at the appropriate times. The pair of triggers will work together to enable and disable the cache over time.

Caching and performance
Introducing a cache will not always improve performance. In some cases, the overhead of reading from the cache may be larger than the overhead of accessing the original data source.

When performance is the goal of using a cache, testing should be performed to ensure that this goal is reached. The proper choice of file versus database caching and use of indexes in a database (as described earlier) may be required to achieve this goal.
How to handle failures
A cache refresh may fail due to an outage or change to the original data source. The following are some options for handling such a failure.

Client errors or stale data:

The default behavior after a refresh failure is to return an error to the client if no data is loaded, or to return the currently loaded (but stale) data to the client.

To make the client always receive errors following a refresh failure, the cache Clear Rule can be set to when refresh fails. This will cause the cache to be cleared and to report errors on all read attempts following the failure.

To make the client tolerate some amount of stale data, both expiration and scheduled refreshing can be combined. For example, if the cache is set to refresh every 4 hours and the expiration period is set to 24 hours, then the data will be allowed to go stale for up to 6 failed refreshes before it gets cleared and starts reporting errors.

Responding to failures:

A Trigger resource can be created that subscribes to cache refresh failures and sends an e-mail or executes a Procedure with appropriate error handling logic. This can be used to alert someone that there is a problem or to possibly perform a corrective task.

Incremental caching

For some situations with very large data sets, it may be impractical to completely refresh the cache data each time. It may be desirable to instead only update the actual cache data that changed.

Incremental caching is not a built-in feature of the product, but is rather something that can be accomplished through the use of the API and additional programmed logic. The following sections describe some options, but there are other ways to accomplish incremental caching.

How to determine what changed

Incremental caching requires that there be some way to query the original data source as to what rows were inserted, updated, and deleted. It also requires that the rows in the data have a primary key that can uniquely identify each row.

The means of determining what rows were inserted, updated, and deleted will vary depending on the source. Some tables include a timestamp column for when the row of data was changed. Other tables may use sequence numbering that can be tracked to determine changes. In more extreme cases, database triggers may be used to detect changes and copy relevant data to a staging table.

In any case, the determination of what changed is not a built-in feature of the Virtual View Manager Server.

Option #1: Transactional database

For this option, a Procedure is written in Java or SQL Script and scheduled to run on a regular basis. This procedure performs some logic to identify what rows have changed. Then, the server ID is acquired using GetProperty('SERVER_ID').
Then, the data source Status Table is queried to find the currently active cachekey. This is in the row with the appropriate server ID, resource path, and status column with value ‘A’.

Then, the appropriate INSERT, UPDATE, and DELETE operations can be performed on the data table using the cachekey as a filter to avoid updating any other keys.

This approach to incremental refresh only works if the data source offers transaction isolation. It relies on having none of the data changes becoming visible to other already open transactions. If the data source does not isolate transactions, then the next option is required.

**Option #2: Non-transactional updates**

For this option, a Procedure is written in Java or SQL Script and scheduled to run on a regular basis. This procedure performs some logic to identify what rows have changed.

Then, CreateResourceCacheKey() is called to generate a new cachekey value.

Then, the server ID is acquired using GetProperty('SERVER_ID').

Then, INSERT a new row into 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.

Then, LoadResourceCacheStatus() is called to inform the server of the in-progress refresh.

Then 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.

Then, 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.

Then, LoadResourceCacheStatus() is called to inform the server of the newly active data.

Since this approach does require all new rows to be created in the cache data table, it is more expensive than the first option, but since the rows can be copied from the previously active data, it is possible to make this perform reasonably well.

---

**Database specifics**

Each database has suggested and allowed native data types for storing cache data. They are described in tables that follow.

In the following tables:
- The Data Type column shows each data type that may display as a projection from a View or Procedure output parameter.
- The Preferred Native Type is the type that will be suggested in the DDL when using the feature to create or recreate tables.
- The Other Allowed Native Types column shows other types in the database that can be used as alternatives to the preferred type. A trailing '+' on a number entry means "or higher".

Table 21. File Cache

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>BIT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>DECIMAL(10+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
<td>DECIMAL(19+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>NUMERIC(p,q)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), CLOB, Integer type with enough resolution</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), CLOB, Integer type with enough resolution</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n)</td>
<td>CHAR(n+), CLOB</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n)</td>
<td>(VARCHAR(n+), CLOB)</td>
</tr>
<tr>
<td>CLOB</td>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BINARY(n)</td>
<td>BINARY(n+), BLOB</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>VARBINARY(n)</td>
<td>VARBINARY(n+), BLOB</td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BOOLEAN</td>
<td>BIT, TINYINT, SMALLINT, INTEGER, BIGINT</td>
</tr>
<tr>
<td>XML</td>
<td>CLOB</td>
<td>VARCHAR(*) [Clips data if column is too small]</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>
### Table 22. IBM DB2

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>SMALLINT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>TINYINT</td>
<td>SMALLINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>DECIMAL(10+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
<td>DECIMAL(19+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>DOUBLE</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>DECIMAL(p,q), CLOB [if p &gt; 31]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), GRAPHIC(p+3+), CLOB</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q), CLOB [if p &gt; 31]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), GRAPHIC(p+3+), CLOB</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) CLOB[if n&gt;254]</td>
<td>CHAR(n+), GRAPHIC(n+), VARCHAR(n+), VARCHARIC(n+), CLOB</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) CLOB[if n&gt;254]</td>
<td>VARCHAR(n+), VARCHARIC(n+), CLOB, LONG_VARGRAPHIC</td>
</tr>
<tr>
<td>CLOB</td>
<td>CLOB</td>
<td>LONG_VARGRAPHIC</td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>VARCHAR(26+)</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>SMALLINT</td>
<td>INTEGER, BIGINT</td>
</tr>
<tr>
<td>XML</td>
<td>CLOB</td>
<td>VARCHAR(<em>) [Clips data if column is too small], VARCHARIC(</em>), LONG_VARGRAPHIC</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 23. Microsoft SQL Server

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>BIT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>Data type</td>
<td>Preferred native type</td>
<td>Other allowed native types</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TINYINT</td>
<td>SMALLINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR(20+), NVARCHAR(20+)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR(20+), NVARCHAR(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>DECIMAL(10+,0), Larger integer type, VARCHAR(20+), NVARCHAR(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
<td>DECIMAL(19+,0), Larger integer type, VARCHAR(20+), NVARCHAR(20+)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>REAL</td>
<td>FLOAT, VARCHAR(24+)</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>FLOAT</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>DECIMAL(p,q), TEXT(if p &gt;38)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), NVARCHAR(p+3+), TEXT, NTEXT</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q) TEXT(if p&gt;38)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), NVARCHAR(p+3+), TEXT, NTEXT</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) TEXT(if p&gt;38)</td>
<td>CHAR(n+), NCHAR(n+), VARCHAR(n+), NVARCHAR(n+), TEXT, NTEXT</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) TEXT ((if n &gt; 255)</td>
<td>VARCHAR(n+), NVARCHAR(n+), TEXT, NTEXT</td>
</tr>
<tr>
<td>CLOB</td>
<td>TEXT</td>
<td>NTEXT</td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BINARY(n) IMAGE (if n&gt; 255)</td>
<td>BINARY(n+), IMAGE</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>VARBINARY(n) IMAGE (if n&gt; 255)</td>
<td>VARBINARY(n+), IMAGE</td>
</tr>
<tr>
<td>BLOB</td>
<td>IMAGE</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>VARCHAR(10)</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>VARCHAR(15)</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATETIME</td>
<td></td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BIT</td>
<td>TINYINT, SMALLINT, INTEGER, BIGINT</td>
</tr>
<tr>
<td>XML</td>
<td>TEXT</td>
<td>VARCHAR(*) [Clips data if column is too small], TEXT</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>
Notes

- The page size limits the number of bytes that can be stored directly in a column. This means it is possible when executing DDL to get an error that the resulting table requires a row size greater than this limit. The solution is to either raise the page size for the database, or to use indirect storage types such as TEXT and IMAGE. Virtual View Manager chooses TEXT and IMAGE types if a value requires more than 255 bytes of storage for this reason, although Microsoft SQL Server does allow VARCHAR and VARBINARY up to 8,000 bytes. Hand tuning of the data types used in a table can be used to improve storage efficiency.

- Microsoft SQL Server TINYINT is range 0 to 255 and Virtual View Manager TINYINT is -128 to 127, so these types are not compatible.

- Microsoft SQL Server DATETIME has accuracy only to within 3.33ms, so some rounding error may occur in the milliseconds.

### Table 24. MySQL

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>BIT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>DECIMAL(10+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
<td>DECIMAL(19+,0), Larger integer type, VARCHAR(20+)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>NUMERIC(p,q) TEXT(if p &gt;30)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), TINYTEXT, MEDIUMTEXT, LONGTEXT, Integer type with enough resolution</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q) TEXT(if p&gt;30)</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), TINYTEXT, MEDIUMTEXT, LONGTEXT, Integer type with enough resolution</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) LONGTEXT(if p&gt;255)</td>
<td>CHAR(n+), TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) LONGTEXT (if n &gt; 255)</td>
<td>VARCHAR(n+), TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT</td>
</tr>
<tr>
<td>CLOB</td>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BLOB LONGBLOB (if n &gt; 255)</td>
<td>TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>BLOB LONGBLOB (if n &gt; 255)</td>
<td>TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB</td>
</tr>
<tr>
<td>BLOB</td>
<td>LONGBLOB</td>
<td></td>
</tr>
</tbody>
</table>
### Table 24. MySQL (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATETIME</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BIT</td>
<td>BIT, BOOL</td>
</tr>
<tr>
<td>XML</td>
<td>LONGTEXT</td>
<td>VARCHAR(*), TINYINT, TEXT, MEDIUMTEXT (Clips data if col too small)</td>
</tr>
</tbody>
</table>

**Notes**

- MySQL removes trailing spaces from strings stored in a VARCHAR column and trailing 0x20 bytes from a VARBINARY column.
- MySQL truncates millisecond data from TIME, DATETIME, and TIMESTAMP columns.
- MySQL changes any NULL stored in a TIMESTAMP column into the current date. Use DATETIME to preserve NULL values.
- Virtual View Manager creates tables using the "utf8" character set to properly handle international characters. You can create the tables using other character sets based on your performance and character set needs.
- Small variations in the least significant digits may be encountered when storing FLOAT and DOUBLE values due to the way the driver handles and database stores such data.

### Table 25. Oracle

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>NUMBER(1, 0)</td>
<td>NUMBER(1+, 0)</td>
</tr>
<tr>
<td>TINYINT</td>
<td>NUMBER(3, 0)</td>
<td>NUMBER(3+, 0), VARCHAR20+, NVARCHAR20+</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>NUMBER(5, 0)</td>
<td>NUMBER(5+, 0), VARCHAR20+, NVARCHAR20+</td>
</tr>
<tr>
<td>INTEGER</td>
<td>NUMBER(10, 0)</td>
<td>NUMBER(10+, 0), VARCHAR20+, NVARCHAR20+</td>
</tr>
<tr>
<td>BIGINT</td>
<td>NUMBER(19, 0)</td>
<td>NUMBER(19+, 0), VARCHAR20+, NVARCHAR20+</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td>VARCHAR24+, FLOAT, BINARY_FLOAT, BINARY_DOUBLE</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>VARCHAR24</td>
<td>VARCHAR24+, FLOAT, BINARY_FLOAT, BINARY_DOUBLE</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>NUMBER(p,q) CLOB [if p&gt; 38]</td>
<td>NUMBER(p+,q+), VARCHAR2(p+ 3+), NVARCHAR2(p+3+), CLOB</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>NUMBER(p,q) CLOB [if p&gt; 38]</td>
<td>NUMBER(p+,q+), VARCHAR2(p+ 3+), NVARCHAR2(p+3+), CLOB</td>
</tr>
</tbody>
</table>
### Table 25. Oracle (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) CLOB [if n &gt; 2000]</td>
<td>CHAR(n+), VARCHAR2(n+), NVARCHAR2(n+), CLOB</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) CLOB [if n &gt; 4000]</td>
<td>VARCHAR2(n+), NVARCHAR2(n+), CLOB</td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
<td>CLOB</td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>RAW(n) BLOB [if n &gt; 255]</td>
<td>RAW(n+), BLOB</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>RAW(n) BLOB [if n &gt; 255]</td>
<td>RAW(n+), BLOB</td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
<td>BLOB</td>
</tr>
<tr>
<td>DATE</td>
<td>VARCHAR2(10)</td>
<td>VARCHAR2(10+), NVARCHAR2(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>VARCHAR2(15)</td>
<td>VARCHAR2(15+), NVARCHAR2(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP(9) FOR 9i AND 10g</td>
<td></td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>NUMBER(1,0)</td>
<td>NUMBER(1+,0)</td>
</tr>
<tr>
<td>XML</td>
<td>CLOB</td>
<td>VARCHAR(<em>), NVARCHAR2(</em>) [Clips data if column is too small]</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>

### Notes

- Oracle changes any empty string stored in a VARCHAR2 or NVARCHAR2 column to NULL. This can have the effect of altering empty string data stored in such columns.
- Oracle FLOAT columns have a maximum of 126 digits, equivalent to a floating point number with exponent "E125". Virtual View Manager FLOAT values have a maximum of E38 and DOUBLE values have a maximum of "E308". This is why VARCHAR is used to store Virtual View Manager DOUBLE values by default. The FLOAT type may be used if your values fit within that range.

### Table 26. Sybase

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>BIT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>TINYINT</td>
<td>SMALLINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR2(20+), NVARCHAR2(20+)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR2(20+), NVARCHAR2(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INT</td>
<td>DECIMAL(10+,0), VARCHAR2(20+), NVARCHAR2(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>DECIMAL(19,0)</td>
<td>DECIMAL(19+,0)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>REAL</td>
<td>FLOAT, VARCHAR2(24+)</td>
</tr>
</tbody>
</table>
### Table 26. Sybase (continued)

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUBLE</td>
<td>FLOAT</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>DECIMAL(p,q) TEXT [if p&gt;38]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), NVARCHAR(p+3+), TEXT</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q) TEXT [if p&gt;38]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), NVARCHAR(p+3+), TEXT</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) TEXT [if n&gt;255]</td>
<td>CHAR(n+), NCHAR(n+), VARCHAR(n+), NVARCHAR(n+), TEXT</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) TEXT [if n&gt;255]</td>
<td>VARCHAR(n+), NVARCHAR(n+), TEXT</td>
</tr>
<tr>
<td>CLOB</td>
<td>TEXT</td>
<td>NTEXT</td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BINARY(n) IMAGE [if n&gt;255]</td>
<td>BINARY(n+), IMAGE</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>VARBINARY(n) IMAGE [if n&gt;255]</td>
<td>VARBINARY(n+), IMAGE</td>
</tr>
<tr>
<td>BLOB</td>
<td>IMAGE</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>VARCHAR(10)</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>VARCHAR(15)</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMPP</td>
<td>VARCHAR(26)</td>
<td>DATETIME, VARCHAR(26+)</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BIT</td>
<td>TINYINT, SMALLINT, INTEGER</td>
</tr>
<tr>
<td>XML</td>
<td>TEXT</td>
<td>VARCHAR(*) [Clips data if column too small]</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>

### Notes
- Sybase's page size limits the number of bytes that can be stored directly in a column. This means it is possible when executing DDL to get an error that the resulting table requires a row size greater than this limit. The solution is to either raise the page size for the database, or to use indirect storage types such as TEXT. Virtual View Manager chooses TEXT types if a value requires more than 255 bytes of storage for this reason, although Sybase does allow VARCHAR and VARBINARY with larger size. Hand tuning of the data types used in a table can be used to improve storage efficiency.
- Sybase DATETIME has accuracy only to within 3.33ms, so some rounding error may occur in the milliseconds.

### Table 27. Teradata

<table>
<thead>
<tr>
<th>Data type</th>
<th>Preferred native type</th>
<th>Other allowed native types</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT</td>
<td>BYTEINT</td>
<td>DECIMAL(1+,0), Larger integer type</td>
</tr>
<tr>
<td>TINYINT</td>
<td>BYTEINT</td>
<td>DECIMAL(3+,0), Larger integer type, VARCHAR(20+), VARGRAPHIC(20+)</td>
</tr>
<tr>
<td>Data type</td>
<td>Preferred native type</td>
<td>Other allowed native types</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
<td>DECIMAL(5+,0), Larger integer type, VARCHAR(20+), VARGRAPHIC(20+)</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>DECIMAL(10+,0), VARCHAR(20+), GRAPHIC(20+), VARGRAPHIC(20+)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>CHAR(20)</td>
<td>DECIMAL(19+,0), VARCHAR(20+)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>FLOAT</td>
<td>VARCHAR(24+)</td>
</tr>
<tr>
<td>NUMERIC(p,q)</td>
<td>DECIMAL(p,q) CLOB [if p&gt;18]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), GRAPHIC(p+3+), CLOB</td>
</tr>
<tr>
<td>DECIMAL(p,q)</td>
<td>DECIMAL(p,q) CLOB [if p&gt;18]</td>
<td>DECIMAL(p+,q+), VARCHAR(p+3+), GRAPHIC(p+3+), CLOB</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n) CLOB [if n&gt;32,000]</td>
<td>CHAR(n+), GRAPHIC(n+), VARCHAR(n+), CLOB</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR(n) CLOB [if n&gt;32,000]</td>
<td>VARCHAR(n+), VARGRAPHIC(n+)</td>
</tr>
<tr>
<td>CLOB</td>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>BINARY(n)</td>
<td>BYTE(n) BLOB [if n&gt;32,000]</td>
<td>BYTE(n+)</td>
</tr>
<tr>
<td>VARBINARY(n)</td>
<td>VARBYTE(n) BLOB [if n&gt;32,000]</td>
<td>VARBYTE(n+)</td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>VARCHAR(10+)</td>
</tr>
<tr>
<td>TIME</td>
<td>VARCHAR(15)</td>
<td>VARCHAR(15+)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>VARCHAR(26+)</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>BYTEINT</td>
<td>SMALLINT, INTEGER</td>
</tr>
<tr>
<td>XML</td>
<td>CLOB</td>
<td>VARCHAR(*) [Clips data if column too small]</td>
</tr>
<tr>
<td>OTHER</td>
<td>(not cacheable)</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 12. Client interfaces

This chapter describes how to retrieve data through a JDBC, ODBC, ADO.NET or SOAP client program.

Client applications can retrieve data only from published resources through five different access mechanisms:

- Interactively, by executing a query in IBM Cognos Virtual View Manager
- From a client program through a JDBC connection
- From a client program through an ODBC connection
- From a client program through an ADO.NET connection
- From a client program through a SOAP connection

Client connectivity information

The IBM Cognos Virtual View Manager server listens to port 9401 (the default port) for JDBC and ODBC connections. By default, the JDBC/ODBC drivers are installed in sub-directories of the installation root directory.

The JDBC driver is installed in the apps\jdbc\lib. The JDBC driver is archived in a .jar file (csjdbc.jar) that must be placed in an appropriate directory of the client application that will be connected to the server.

The ODBC driver is installed in apps\odbc\lib. The ODBC driver has its own installation programs (VVMOdbcInstall101.exe for the Windows platform and driverConfig for UNIX platforms) for connecting the ODBC client software to the server.

You can install the JDBC and ODBC drivers independently from the installation CD onto client machines.

If Virtual View Manager Server is running on a machine that has a Windows firewall, no JDBC/ODBC client can connect to the server via unless the firewall is configured to allow such connectivity.

Connections via JDBC

IBM Cognos Virtual View Manager data services are accessible through a standard JDBC interface.

The JDBC driver, which is archived in the csjdbc.jar file, is available in the <installation_directory>\apps\jdbc\lib directory:

Setting the CLASSPATH for csjdbc.jar

To use the JDBC driver, make sure that csjdbc.jar is available in your system's CLASSPATH. You must set the CLASSPATH for csjdbc.jar as described here.

Procedure

1. If your computer is running on the Microsoft Windows operating system, run the following commands to set the CLASSPATH and run a program:
set CLASSPATH=%CLASSPATH%;<PATH>\csjdbc.jar
java -classpath "%CLASSPATH%;<PATH>\csjdbc.jar"
<JDBC_PROGRAM>

<Path> is the valid path where the JDBC driver is located.

2. If your computer is running on the UNIX operating system, run the following commands to set the CLASSPATH and run a program:

export CLASSPATH=$CLASSPATH:<PATH>/csjdbc.jar
java -classpath "$CLASSPATH:<PATH>/csjdbc.jar"
<JDBC_PROGRAM>

<Path> is the valid path where the JDBC driver is located.

Properties for the driver connection URL

The following table lists the names of properties that can be specified in the JDBC connection URL.

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaseSensitive</td>
<td>Specifies case sensitivity in the request values. Requests are not case sensitive by default. Default: false</td>
</tr>
<tr>
<td>connectTimeout</td>
<td>Time-out for initial connection, in seconds: 30 (for example) 0 (zero, for infinite time-out)</td>
</tr>
<tr>
<td>encrypt</td>
<td>Set encrypt equal to true and the JDBC messages will be automatically passed to the SSL port (default is 9403) for processing with the Virtual View Manager SSL Certificate.</td>
</tr>
<tr>
<td>fetchRows</td>
<td>Maximum number of rows to fetch for batching purposes: 500 (for example) 0 (zero, for infinite number of rows)</td>
</tr>
<tr>
<td>fetchBytes</td>
<td>Maximum number of rows to fetch for batching purposes based on batch size: 128K (for example) 0 (zero, for infinite number of bytes)</td>
</tr>
<tr>
<td>ignoreTrailingSpace</td>
<td>Ignores trailing spaces following values. Default: false</td>
</tr>
<tr>
<td>requestTimeout</td>
<td>Time-out for query commands and other requests</td>
</tr>
</tbody>
</table>
| paramMode      | Controls the behavior of OUT parameters for stored procedures. paramMode uses one of the following four attributes:  
|                | • normal — OUT parameters reported in procedure metadata as OUT parameters  
|                | • return — OUT parameters are reported as return values  
|                | • omit — OUT parameters omitted from metadata  
|                | • omitCursors — Output cursors omitted from metadata                                                                                       |
**Multiple credentials for JDBC connection**

The method `setDataSourceCredentials()` can be used to register multiple pass-through credentials for the current client session.

This method takes three string arguments: data source path (nullable), user name, and password. The `setDataSourceCredentials` method may be called many times if connections must be established with multiple data sources enabled with pass-through.

When a data source requires pass-through credentials, a client program will try to access the data source in the following order:

- If credentials for accessing the data source are set using the `setDataSourceCredentials` method, the program will attempt to use those credentials to secure access to the data source. If the attempt fails, the program will fail.
- If the path to the data source is specified as NULL and valid user name and password are available in the `setDataSourceCredentials` method, the program will try to access the data source. If the attempt fails, the program will fail.
- If the previous two attempts fail, the program will try to access the data source using the same login credentials used to establish a connection with the Virtual View Manager server.

**About clearing data source credentials**

When using pooled connections, JDBC clients should clear any credentials that were set prior to closing the connection by calling the `clearAllDataSourceCredentials` method. If the client connection is terminated abnormally, the connection thread is not returned to the pool for use by other clients.

The `clearAllDataSourceCredentials` method does not have arguments.
**Code sample template**

This section provides a sample template for you to get started using the JDBC driver in your Java code.

```java
import java.sql.*;

class JdbcSample {
    public static void main(String args[]) {
        if (args.length != 7) {
            System.err.println("usage : prog <datasource name> <host name> <port> <user> <password> <domain name> \"<sql statement>\"\n");
            System.exit(1);
        }
        String datasource = args[0]; // datasource_name
        String ip = args[1]; // IP or host name of Virtual View Manager Server
        int port = 0;
        try {
            port = Integer.parseInt(args[2]);
        } catch (Exception e) {
            port = 9401;
        }
        String userName = args[3];
        String password = args[4];
        String domain = args[5];
        String url = null;
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        ResultSetMetaData rsmd = null;
        try {
            Class.forName("cs.jdbc.driver.CompositeDriver");
            url = "jdbc:cognos:dbapi@" + ip + ":" + port + ":" + domain + ".datasource='" + datasource + "'";
            conn = DriverManager.getConnection(url, userName, password);
            ((cs.jdbc.driver.CompositeConnection)conn).clearAllDataSourceCredentials();
            ((cs.jdbc.driver.CompositeConnection)conn).setDataSourceCredentials(datasource, userName, password);
            stmt = conn.createStatement();
            boolean isNotUpdate = stmt.execute(args[6]);
            int rows = 0;
            // return type is a result set
            if (isNotUpdate == true) {
                rs = stmt.getResultSet();
                if (rs == null) {
                    throw new SQLException("sql='"+args[6]+"' did not generate a result set");
                }
            }
            rsmd = rs.getMetaData();
            int columns = rsmd.getColumnCount();
            System.out.println("column count = "+ columns);
            rows = 1;
            int type = 0;
            while (rs.next()) {
                System.out.print("row = " + rows + " " );
                for (int i=1; i <= columns; i++) {
                    type = rsmd.getColumnType(i);
                    switch (type) {
                        case Types.INTEGER:
                            System.out.print(" col[" + i + "]=");
                            break;
                        case Types.SMALLINT:
                            System.out.print(" col[" + i + "]=");
                            break;
                    }
                    rs.getInt(i) + " ");
                }
            }
        }
    }
}
```

---

*226  IBM Cognos Virtual View Manager Version 10.2.0: User Guide*
+ rs.getShort(i) + " ";
                break;
        case Types.TINYINT:
                System.out.print(" col[" + i + "]="
                break;
        case Types.BIGINT:
                System.out.print(" col[" + i + "]="
                break;
        case Types.FLOAT:
                System.out.print(" col[" + i + "]="
                break;
        case Types.REAL:
                System.out.print(" col[" + i + "]="
                break;
        case Types.DECIMAL:
                System.out.print(" col[" + i + "]="
                break;
        case Types.DOUBLE:
                System.out.print(" col[" + i + "]="
                break;
        case Types.NUMERIC:
                System.out.print(" col[" + i + "]="
                break;
        case Types.CHAR:
                System.out.print(" col[" + i + "]="
                break;
        case Types.VARCHAR:
                System.out.print(" col[" + i + "]="
                break;
        case Types.LONGVARCHAR:
                System.out.print(" col[" + i + "]="
                break;
        case Types.DATE:
                System.out.print(" col[" + i + "]="
                break;
        case Types.TIME:
                System.out.print(" col[" + i + "]="
                break;
        case Types.TIMESTAMP:
                System.out.print(" col[" + i + "]="
                break;
        case Types.BOOLEAN:
                System.out.print(" col[" + i + "]="
                break;
        default:
                System.out.print(" col[" + i + "]="
                break;
        }
    }
    rows++;
Supported JDBC methods

The IBM Cognos Virtual View Manager JDBC driver fully supports the JDBC connection pooling as well as the following JDBC methods from DatabaseMetaData and ResultSet.

Methods in class DatabaseMetaData

getBestRowIdentifier(String, String, String, int, boolean)
getColumnPrivileges(String, String, String)
getCrossReference(String, String, String, String, String, String)
getExportedKeys(String, String, String)
getImportedKeys(String, String, String)
getIndexInfo(String, String, String, boolean, boolean)
getPrimaryKeys(String, String, String)
getTablePrivileges(String, String, String)
getVersionColumns(String, String, String)
Fully supported methods in DatabaseMetaData

getCatalogs()
getColumns(String, String, String, String)
getProcedureColumns(String, String, String, String)
getProcedures(String, String, String)
getSchemas()
getTables(String, String, String, String[])

Fully supported methods in ResultSet

getBigDecimal(int)
getBigDecimal(String)
getBoolean(int)
getBoolean(String)
getByte(int)
getByte(String)
g getDate(int)
g getDate(String)
g getDouble(int)
g getDouble(String)
g getFloat(int)
g getFloat(String)
g getInt(int)
g getInt(String)
g getLong(int)
g getLong(String)
g getObject(int)
g getObject(String)
g getShort(int)
g getShort(String)
Unsupported methods

If you try to access a JDBC method that is not supported in Virtual View Manager, the system will show an error.

You can disable this by setting the value of unsupportedMode to be silent in the JDBC connection URL, as in the following example:

jdbc:cognos:dbapi@localhost9401?domain=cognos&dataSource=examples
&unsupportedMode=silent

The following methods are not supported.

callablestatement

void registerOutParameter(int parameterIndex, int sqlType, int scale)

BigDecimal getBigDecimal(int parameterIndex, int scale)

Object getObject(int parameterIndex, Map map)

Ref getRef(int parameterIndex)

Array getArray(int parameterIndex)

Date getDate(int parameterIndex, Calendar cal)

Time getTime(int parameterIndex, Calendar cal)

Timestamp getTimestamp(int parameterIndex, Calendar cal)

void registerOutParameter(int parameterIndex, int sqlType, String typeName)

void registerOutParameter(String parameterName, int sqlType, int scale)

void registerOutParameter(String parameterName, int sqlType, String typeName)

URL getURL(int parameterIndex)

void setURL(String parameterName, URL x)

void setAsciiStream(String parameterName, InputStream x, int length)

void setBinaryStream(String parameterName, InputStream x, int length)
void setCharacterStream(String parameterName, Reader reader, int length)

setObject(String parameterName, Object x, int targetSqlType, int scale)

void setDate(String parameterName, java.sql.Date x, Calendar cal)
void setTime(String parameterName, java.sql.Time x, Calendar cal)

void setTimestamp(String parameterName, java.sql.Timestamp x, Calendar cal)
void setNull(String parameterName, int sqlType, String typeName)

Object getObject(String parameterName, Map map)
Ref getRef(String parameterName)
Array getArray(String parameterName)
URL getURL(String parameterName)

**Connection**

void setReadOnly(boolean readOnly)

void setCatalog(String catalog)
void setTypeMap(Map map)

String nativeSQL(String sql)
void setHoldability(int holdability)

Savepoint setSavepoint()

setSavepoint(String name)

void releaseSavepoint(Savepoint savepoint)

PreparedStatement prepareStatement(String sql, int autoGeneratedKeys)
PreparedStatement prepareStatement(String sql, int[] columnIndexes)
PreparedStatement prepareStatement(String sql, String[] columnNames)

**DatabaseMetaData**

boolean allTablesAreSelectable()

boolean allProceduresAreCallable()

boolean supportsConvert(int fromType, int toType)

ResultSet getSuperTypes(String catalog, String schemaPattern, String typeNamePattern)

ResultSet getSuperTables(String catalog, String schemaPattern, String typeNamePattern)
ResultSet getAttributes(String catalog, String schemaPattern, String typeNamePattern, String attributeNamePattern)

int getSQLStateType()

**PreparedStatement**

int executeUpdate(String sql)

void setAsciiStream(int parameterIndex, InputStream x, int length)

void setUnicodeStream(int parameterIndex, InputStream x, int length)

void setBinaryStream(int parameterIndex, InputStream x, int length)

void setObject(int parameterIndex, Object x, int targetSqlType, int scale)

boolean execute(String sql)

void addBatch()

void setCharacterStream(int parameterIndex, Reader reader, int length)

void setRef(int i, Ref x)

setArray(int i, Array x)

void setDate(int parameterIndex, java.sql.Date x, Calendar cal)

void setTime(int parameterIndex, java.sql.Time x, Calendar cal)

void setTimestamp(int parameterIndex, java.sql.Timestamp x, Calendar cal)

void setNull(int paramIndex, int sqlType, String typeName)

void setURL(int parameterIndex, URL x)

**Blob**

int setBytes(long pos, byte[] bytes)

int setBytes(long pos, byte[] bytes, int offset, int len)

java.io.OutputStream setBinaryStream(long pos)

**Clob**

int setString(long pos, String str)

int setString(long pos, String str, int offset, int len)

java.io.OutputStream setAsciiStream(long pos)

java.io.Writer setCharacterStream(long pos)
**ResultSet**

BigDecimal getBigDecimal(int columnIndex, int scale)

InputStream getAsciiStream(int columnIndex)

InputStream getUnicodeStream(int columnIndex)

String getCursorName()

void beforeFirst()

void afterLast()

boolean first()

boolean last()

boolean absolute(int row)

boolean relative(int rows)

boolean previous()

void setFetchDirection(int direction)

void updateNull(int columnIndex)

void updateBoolean(int columnIndex, boolean x)

void updateByte(int columnIndex, byte x)

void updateShort(int columnIndex, short x)

void updateInt(int columnIndex, int x)

void updateLong(int columnIndex, long x)

void updateFloat(int columnIndex, float x)

void updateDouble(int columnIndex, double x)

void updateBigDecimal(int columnIndex, BigDecimal x)

void updateString(int columnIndex, String x)

void updateBytes(int columnIndex, byte x[])

void updateDate(int columnIndex, java.sql.Date x)

void updateTime(int columnIndex, java.sql.Time x)

void updateTimestamp(int columnIndex, java.sql.Timestamp x)

void updateAsciiStream(int columnIndex, InputStream x, int length)
void updateBinaryStream(int columnIndex, InputStream x, int length)
void updateCharacterStream(int columnIndex, Reader x, int length)
void updateObject(int columnIndex, Object x, int scale)
void updateObject(int columnIndex, Object x)
void updateNull(String columnName)
void updateBoolean(String columnName, boolean x)
void updateByte(String columnName, byte x)
void updateShort(String columnName, short x)
void updateInt(String columnName, int x)
void updateLong(String columnName, long x)
void updateFloat(String columnName, float x)
void updateDouble(String columnName, double x)
void updateBigDecimal(String columnName, BigDecimal x)
void updateString(String columnName, String x)
void updateBytes(String columnName, byte x[])
void updateDate(String columnName, java.sql.Date x)
void updateTime(String columnName, java.sql.Time x)
void updateTimestamp(String columnName, java.sql.Timestamp x)
void updateAsciiStream(String columnName, InputStream x, int length)
void updateBinaryStream(String columnName, InputStream x, int length)
void updateCharacterStream(String columnName, Reader reader, int length)
void updateObject(String columnName, Object x, int scale)
void updateObject(String columnName, Object x)
void insertRow()
void updateRow()
void deleteRow()
void refreshRow()
void cancelRowUpdates()
void moveToInsertRow()
void moveToCurrentRow()
Object getObject(int i, Map map)
Ref getRef(int i)
Array getArray(int i)
Object getObject(String colName, Map map)
Ref getRef(String colName)
Array getArray(String colName)
Date getDate(int columnIndex, Calendar cal)
Date getDate(String columnName, Calendar cal)
Time getTime(int columnIndex, Calendar cal)
Time getTime(String columnName, Calendar cal)
Timestamp getTimestamp(int columnIndex, Calendar cal)
Timestamp getTimestamp(String columnName, Calendar cal)
URL getURL(int columnIndex)
URL getURL(String columnName)
void updateRef(int columnIndex, Ref x)
void updateRef(String columnName, Ref x)
void updateBlob(int columnIndex, Blob x)
void updateBlob(String columnName, Blob x)
void updateClob(int columnIndex, Clob x)
void updateClob(String columnName, Clob x)
void updateArray(int columnIndex, Array x)
void updateArray(String columnName, Array x)

**Statement**

void getWarnings()
void clearWarnings()
void setMaxFieldSize(int max)
void setEscapeProcessing(boolean enable)
void addBatch(String sql)
Statement types

The Virtual View Manager JDBC driver currently supports the statement type
Statements and Prepare Statements.

To use the examples given here in your environment, supply the appropriate
values for the following, as relevant:
  • Login credentials for accessing the server
  • Login credentials for accessing the data source
  • SELECT statement

When you compile and run your code, set the CLASSPATH to csjdbc.jar, which
contains the Virtual View Manager JDBC driver.

Statements

This section contains examples to illustrate the use of statements.

Example 1

This example demonstrates how to submit a simple SELECT select statement to the
Virtual View Manager JDBC driver.

Here, the data source does not require pass-through login credentials.

Login credentials used here for accessing the server are compUser and
compPassword. These login credentials, connection properties, and the query
statement are rendered in bold type.

```java
import java.util.*;
import java.sql.*;
public class SelectExample {
    public static void main(String[] arg) throws Exception {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
```

```java
            // SQL SELECT statement
            String sql = "SELECT * FROM table1 WHERE column1 = ?";

            // Set auto-generated keys
            int autoGeneratedKeys = ResultSet.TYPE_FORWARD_ONLY;

            // Prepare the statement
            PreparedStatement pstmt = conn.prepareStatement(sql, autoGeneratedKeys);

            // Set parameters
            pstmt.setInt(1, 101);

            // Execute the statement
            rs = pstmt.executeQuery();

            // Process ResultSet
            while (rs.next()) {
                // Example of processing data
                System.out.println(rs.getString("column2"));
            }
        }
    }
}
```
Example 2

This example illustrates how to submit a simple SELECT statement to a data source that requires pass-through credentials (dsUser,dsPassword). The login credentials for accessing the Virtual View Manager server (compUser,compPassword) are different from the ones used for accessing the data source.

The setDataSourceCredentials method is used to register the data source pass-through credentials for the current session. This method takes three String arguments representing the path to the datasource, username, and password, as in the following example:

```
setDataSourceCredentials("/shared/sources/dsPassThru","dsUser","dsPassword");
```

Notice that the full path to the data source is given as the first argument for setDataSourceCredentials.

- If the path refers to a data source that a user cannot access (due to a lack of access privileges), an error will be returned by the program.
- Only one user name/password pair can be set for a path.
ResultSet rs = null;
try {
    String url = "jdbc:cognos:dbapi@localhost:9401?"
    +"domain=cognos&dataSource=cdspt";
    String user = "compUser";
    String pass = "compPassword";
    // Load driver
    Class.forName("cs.jdbc.driver.CompositeDriver");
    // Create connection
    conn = DriverManager.getConnection(url, user, pass);
    ((cs.jdbc.driver.CompositeConnection)conn)
        .setDataSourceCredentials("/shared/sources/dsPassThru",
        "dsUser",
        "dsPassword");
    // Create statement
    stmt = conn.createStatement();
    // Execute statement
    rs = stmt.executeQuery("SELECT * FROM catalog.schema.table");
    // Get column count
    ResultSetMetaData rsmd = rs.getMetaData();
    int columns = rsmd.getColumnCount();
    // Get results
    while(rs.next()) {
        for (int i=0; i<columns; i++) {
            Object o = rs.getObject(i+1);
            if (o == null) {
                System.out.print("[NULL]");
            } else {
                System.out.print(o.toString());
            }
            System.out.print(" ");
        }
        System.out.println();
    }
} finally {
    if (rs != null) {
        rs.close();
    }
    if (stmt != null) {
        stmt.close();
    }
    if (conn != null) {
        conn.close();
    }
}

Example 3

This example is similar to the previous one. However, notice that the path to the
data source is specified as NULL. When NULL is specified as the resource path,
the credential is added to the session's list of generic credentials for the user.

The program will try to connect with the data source using different credentials for
the user, but will achieve a successful connection only with a data source that has
the specified user name and password. By not having to specify a resource path,
the client can be ignorant of data source namespace, at the cost of having to try
various login credentials in order to achieve one successful connection.

import java.util.*;
import java.sql.*;
public class multiPassThruWithNull {
    public static void main(String[] arg) throws Exception {
        Connection conn = null;
Example 4

In this example, the login credentials for accessing the Virtual View Manager server are the same as those for accessing the data source.

```java
import java.util.*;
import java.sql.*;
public class multiPassThruWithCISLogInCred {
    public static void main(String[] arg) throws Exception {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            String url = "jdbc:cognos:dbapi@localhost:9401?"
                        + "domain=cognos&dataSource=cdspt";
            String user = "dsUser";
            String pass = "dsPassword";
            // Load driver
            Class.forName("cs.jdbc.driver.CompositeDriver");
            // Create connection
            conn = DriverManager.getConnection(url, user, pass);
            ((cs.jdbc.driver.CompositeConnection)conn)
                .setDataSourceCredentials(NULL, "dsUser", "dsPassword");
            // Create statement
            stmt = conn.createStatement();
            // Execute statement
            rs = stmt.executeQuery("SELECT * FROM catalog.schema.table");
            // Get column count
            ResultSetMetaData rsmd = rs.getMetaData();
            int columns = rsmd.getColumnCount();
            // Get results
            while(rs.next()) {
                for (int i=0; i<columns; i++) {
                    Object o = rs.getObject(i+1);
                    if (o == null) {
                        System.out.print("[NULL]\n");
                        } else {
                        System.out.print(o.toString());
                        } System.out.print(" ");
                System.out.println();
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
        }
    }
}
```
Class.forName("cs.jdbc.driver.CompositeDriver");
// Create connection
conn = DriverManager.getConnection(url, user, pass);
((cs.jdbc.driver.CompositeConnection)conn)
    .setDataSourceCredentials("/shared/sources/dsPassThru", "dsUser", "dsPassword");
// Create statement
stmt = conn.createStatement();
// Execute statement
rs = stmt.executeQuery("SELECT * FROM catalog.schema.table");
// Get column count
ResultSetMetaData rsmd = rs.getMetaData();
int columns = rsmd.getColumnCount();
// Get results
while(rs.next()) {
    for (int i=0; i<columns; i++) {
        Object o = rs.getObject(i+1);
        if (o == null) {
            System.out.print("[NULL]");
        } else {
            System.out.print(o.toString());
        }
        System.out.print(" ");
    }
    System.out.println();
}
} finally {
    if (rs != null) {
        rs.close();
    }
    if (stmt != null) {
        stmt.close();
    }
    if (conn != null) {
        conn.close();
    }
}

Prepared statements

A prepared statement is an object that contains an SQL statement that may have input parameters. This object can then be executed multiple times much more efficiently than preparing and issuing the same statement each time it is needed.

This approach saves query processing time and enhances server performance. Prepared statements can be used any number of times, each time with different parameters as necessary. Use a question mark symbol (?) as a placeholder for a parameter within the SQL statement. Once all the placeholder parameters are set, the query is executed.

For further details on prepared statements, refer to any standard reference book on JDBC API and/or the information provided on JDBC at the SUN Microsystems website: http://java.sun.com/products/jdbc.

Virtual View Manager JDBC-Specific Details

Note that the following rules apply when you submit a prepared statement to the Virtual View Manager JDBC driver to access the server:
• Create and use multiple prepared statements on one connection to the server.
• The server maintains a cache of prepared statements some of which exist across multiple connections, so that when you create a prepared statement and that prepared statement is already in the cache, the server will not need to recreate the query plan. The cache size is controlled by the configuration setting, which you can access via Virtual View Manager Administration. For details on configuration settings, see the “Configuration” chapter in the IBM Cognos Virtual View Manager Administration Guide.

• The placeholder for a query parameter can be used anywhere where a literal can be used.

• Currently, the DatabaseMetaData.getMetaData() method is not supported. However, you can get the ResultSetMetaData by using ResultSet.getMetaData().

The following examples illustrate the usage of prepared statements.

Example 1

The following sample code demonstrates the usage of a prepared statement that contains a simple SELECT statement. In this example, the SELECT statement queries the customers table and retrieves the required data under a certain condition, which initially has the question mark (?) placeholder parameter. This example uses a for loop to set values for parameters in the prepared statement. See the section marked in bold.

```java
import java.sql.*;

public class PreparedStatementSample
{
    private static final String VVM_URL =
        "jdbc:cognos:dbapi@localhost:9401?domain=cognos&dataSource=cubs";
    private static final String VVM_DRIVER =
        "cs.jdbc.driver.CompositeDriver";
    private static final String VVM_USER = "admin";
    private static final String VVM_PASSWORD = "admin";
    public static void main(String[] args) {
        try {
            Class.forName(VVM_DRIVER);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            return;
        }
        try {
            execute();
        } catch (SQLException ex) {
            ex.printStackTrace();
            return;
        }
    }
    private static void execute() throws SQLException {
        Connection conn = DriverManager.getConnection(
            VVM_URL, VVM_USER, VVM_PASSWORD);
        PreparedStatement stmt = conn.prepareStatement(
            "SELECT * FROM products WHERE ProductID = ?");
        for (int i = 1; i <= 5 ; i++) {
            stmt.setInt(1, i);
            ResultSet rs = stmt.executeQuery();
            System.out.println("Row " + i);
            printResultSet(rs);
            rs.close();
            stmt.close();
            conn.close();
        }
    }
}
```
Example 2

The following sample code demonstrates the usage of a prepared statement that contains an INSERT statement. See the section marked in bold. This example works similarly to Example 1, except that here executeUpdate() is used instead of executeQuery() to execute the SQL and the result set is the number of rows affected by the insert operation.

```java
import java.sql.*;
import java.math.BigDecimal;
public class PreparedStatementInsert {
    private static final String VVM_URL = "jdbc:cognos:dbapi@localhost:9401?domain=cognos&dataSource=tutorial";
    private static final String VVM_DRIVER = "cs.jdbc.driver.CompositeDriver";
    private static final String VVM_USER = "admin";
    private static final String VVM_PASSWORD = "admin";
    public static void main(String[] args) {
        try {
            Class.forName(VVM_DRIVER);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            return;
        }
        try {
            execute();
        } catch (SQLException ex) {
            ex.printStackTrace();
            return;
        }
    }
    private static void execute() throws SQLException {
        Connection conn = DriverManager.getConnection(VVM_URL, VVM_USER, VVM_PASSWORD);
        PreparedStatement stmt = conn.prepareStatement("INSERT INTO products (ProductID, ProductName, UnitPrice) VALUES (?, ?, ?)");
        stmt.setInt(1, 50);
        stmt.setString(2, "new");
        stmt.setBigDecimal(3, new BigDecimal(50.00));
        int rowsInserted = stmt.executeUpdate();
        System.out.println("Rows inserted " + rowsInserted);
        stmt.close();
        conn.close();
    }
}
```
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));
        }
    }
}

Example 3

The following sample code demonstrates the usage of a prepared statement that contains an UPDATE statement. See the section marked in bold. This example works similar to Example 2. Here, the result set is the number of rows affected by the update operation.

```java
import java.sql.*;
import java.math.BigDecimal;
public class PreparedStatementUpdate {
    private static final String VVM_URL = "jdbc:cognos:dbapi@localhost:9401?domain=cognos&dataSource=tutorial";
    private static final String VVM_DRIVER = "cs.jdbc.driver.CompositeDriver";
    private static final String VVM_USER = "admin";
    private static final String VVM_PASSWORD = "admin";
    public static void main(String[] args) {
        try {
            Class.forName(VVM_DRIVER);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            return;
        }
        try {
            execute();
        } catch (SQLException ex) {
            ex.printStackTrace();
            return;
        }
        private static void execute() throws SQLException {
            Connection conn = DriverManager.getConnection(VVM_URL, VVM_USER, VVM_PASSWORD);
            PreparedStatement stmt = conn.prepareStatement("UPDATE products SET ProductName = ? WHERE ProductID = ?");
            stmt.setString(1, "newProduct");
            stmt.setBigDecimal(2, new BigDecimal(50.00));
            int rowsUpdated = stmt.executeUpdate();
            System.out.println("Rows updated " + rowsUpdated);
            stmt.close();
            conn.close();
        }
        private static void printResultSet(ResultSet rs) throws SQLException {
            ResultSetMetaData metaData = rs.getMetaData();
            int rowIndex = 0;
            int rowIndex = 0;
            for (int i=1; i<=metaData.getColumnCount(); i++) {
                System.out.println(" Column " + i + " " + metaData.getColumnName(i) + " " + rs.getString(i));
            }
        }
```
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));
    }
}

Example 4

The following sample code demonstrates the usage of a prepared statement that contains a DELETE statement. See the section marked in bold. This example works similar to Example 2. Here, the result set is the number of rows affected by the delete operation.

```java
import java.sql.*;
import java.math.BigDecimal;
public class PreparedStatementDelete {

    private static final String VVM_URL = "jdbc:cognos:dbapi=localhost:9401?domain=cognos&dataSource=tutorial";
    private static final String VVM_DRIVER = "cs.jdbc.driver.CompositeDriver";
    private static final String VVM_USER = "admin";
    private static final String VVM_PASSWORD = "admin";

    public static void main(String[] args) {
        try {
            Class.forName(VVM_DRIVER);
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            return;
        }

        try {
            execute();
        } catch (SQLException ex) {
            ex.printStackTrace();
            return;
        }

    }

    private static void execute() throws SQLException {
        Connection conn = DriverManager.getConnection(VVM_URL, VVM_USER, VVM_PASSWORD);
        PreparedStatement stmt = conn.prepareStatement("DELETE FROM products WHERE ProductID = ?");
        stmt.setInt(1, 50);
        int rowsDeleted = stmt.executeUpdate();
        System.out.println("Rows deleted " + rowsDeleted);
        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));
            }
        }
    }
}
```
Connection to IBM Cognos Virtual View Manager Server using ODBC

32-bit ODBC client applications may connect to the Virtual View Manager server to retrieve published data from services managed, secured, and published by Virtual View Manager defined resources.

Virtual View Manager data services are accessible through industry standard ODBC driver managers. ODBC access to the server requires that a suitable ODBC driver manager be installed on the client machine.

The Virtual View Manager ODBC driver conforms to ODBC 3.5 specification level.

Configuring an ODBC data source depends on the driver manager being used.

ODBC client driver installation requirements

ODBC Client applications may utilize a 32-bit Virtual View Manager driver to connect with the server. The computer on which the ODBC client application resides must have one of the Virtual View Manager drivers to properly connect with the server.

Run one of the VVMOdbcInst101.exe executable to install the vvm84.dll ODBC driver on the client computer.

Data services preparation

No special configuration of is necessary for clients to connect with the server using 32-bit ODBC drivers.

To support any client regardless of the type, create a Virtual View Manager Data Service data source and catalog before configuring the ODBC client to use the 32-bit ODBC driver. The published data service and catalog are utilized as targets.

Publish integrated views, procedures, and web services and JDBC/ODBC application clients will be able to utilize them as if they were from a single data source using the Virtual View Manager Data Services node. Optionally that data service should have a catalog, (though it is not required) for use with the ODBC driver.

Note the configured port number of the JDBC/ODBC port for HTTP traffic. The JDBC/ODBC port is always one greater than the web services HTTP base port setting. SSL is not currently supported by the ODBC driver.

Windows System Data Source Names (DSN) configuration

ODBC clients need a configured DSN using the appropriate ODBC driver on the client computer to enable communications with published Virtual View Manager data services.
By default the ODBC driver is installed in the installation_location\apps\odbc directory; however, each computer hosting ODBC client applications must install the Virtual View Manager ODBC driver to enable connection with Virtual View Manager data services.

The DSN is configured with the appropriate driver as described in the next section.

Client applications may use the same System DSN to connect with Virtual View Manager data services. A User DSN could be configured, but that connection would be available to only a single user on the local machine.

**Before adding a system DSN for an ODBC client**

If ODBC client end-users are prohibited from writing to the Windows directory, then an extra step must be taken to set a system variable that will redirect the ODBC Driver to a directory that will allow creation of a file by the user. During the creation of a new DSN, the Virtual View Manager ODBC driver creates a file named vvmXX.xml to save the DSN configuration settings.

Set a new system variable with a name of VVM_DSN_DIR and with a value of some user modifiable directory.

**Adding a new System DSN**

This section describes creation of a new System DSN, so that ODBC clients connecting to the host get pointed to the proper data source published by Virtual View Manager Data Services.

**Procedure**

1. Open the ODBC Data Source Administrator from Control Panel > Windows Administration Tools > Data Sources (ODBC) to add a new System DSN enabling recognition of the specified data source and catalog.
2. Click the System DSN tab.
3. Click Add to create a new data source and select the Virtual View Manager driver.
4. Click Finish and the Driver Configuration window is displayed.
5. Configure the Virtual View Manager ODBC driver.
   - **DSN Name**—The DSN Name should be some meaningful string that will be used by the client to address the data source.
   - **VVM Host**—This value would be localhost if it were for a test client installed on the same computer as the server, or more conventionally the value will be a DNS designated name or an IP octet.
   - **Port**—Use the number of the JDBC/ODBC HTTP port (default value: 9401).
   - **User Name / Password / Domain**—These fields refer to the Virtual View Manager user name, password, and domain of the user connecting from the client. This connection user profile should match a profile in either the cognos domain or a configured LDAP domain unless anonymous login or dynamic domain login is enabled. The connection profile does not give additional rights or privileges to other users who might use that same connection.
   - **Datasource**—Data source refers to the Virtual View Manager database name published in the Virtual View Manager Data Services node.
   - **Catalog**—Each data source can publish one or many catalogs. Use the Refresh button when all other fields are specified and select one.
6. Test the connection and click OK when finished with the configuration.
If the connection is not yielding results, check the firewall to open the designated port for sending and retrieving messages.

With these configurations both the Virtual View Manager Server and the ODBC client should be ready for use.

**Note:** SQL statements generated by ODBC clients must enclose reserved keywords in double-quotes when they are used as column aliases to get proper results. For example an Excel spreadsheet imports data by sending a MS-Query to a published data source:

```
SELECT VendorDetails."Language" FROM VendorDetails
```

Where language is a reserved keyword and also a column name in the VendorDetails table. A consolidated list of reserved keywords appears at the end of Chapter 1 of the *IBM Cognos Virtual View Manager Reference Guide*. One may work around the difficulty of editing an auto-generated MS-Query by renaming a column name with an alias.

---

**Connection using the ADO.NET driver**

IBM Cognos Virtual View Manager supports native ADO.NET driver functionality on Windows operating systems. The .NET Data provider is written in managed C# code and provides a native implementation of ADO.NET API.

Customers may now use Microsoft Visual Studio 2005 and other software to develop solutions using resources defined by Virtual View Manager through the ADO.NET driver interface. Developers may use Microsoft Visual Studio 2005 with ADO.NET.

The Virtual View Manager ADO.NET driver can install on 32-bit Windows operation systems. The 32-bit client application can connect to the server host ports to retrieve published data from services managed, secured, and published by Virtual View Manager defined resources.

**Installation prerequisite**

The Virtual View Manager ADO.NET driver requires the .NET Framework 2.0 to be installed on the host.

Virtual View Manager supports communication and use with the Visual Studio 2005 edition supplied with an installation of the Standard, Professional, or the Team Developer versions.

Other editions of Visual Studio 2005 are not supported.

The Virtual View Manager ADO.NET native driver installation is delivered as a MSI package.

**Note:** The Windows Installer system requires the latest version available with an installation of Windows XP or Windows Server 2003. Upgrade the Windows Installer system to the latest version.

**Installation**

The Virtual View Manager ADO.NET driver installation package for Windows operating systems is named ADONetInstall.msi.
If Visual Studio 2005 was running during installation of the Virtual View Manager ADO.NET driver, then it will have to be restarted to recognize the new data provider.

**Configuration of connection in Visual Studio 2005**

To communicate with a published Virtual View Manager data service, provide the ADO.NET driver with the appropriate properties for Host, Port, User, Password, and Domain to create a connection.

A 32-bit Windows OS can only use the 32-bit ADO.NET Virtual View Manager driver. Configure the connection between Visual Studio 2005 and Virtual View Manager with the appropriate driver as described in the next section.

**Adding new connections**

Create a new connection so that ADO.NET clients connect with the proper data source published by the Virtual View Manager Data Services host.

**Procedure**

1. Open the Visual Studio 2005 Server Explorer pane by choosing the Server Explorer selection from the View menu.
2. Right click the Data Connections node and select the option to Add Connection. The Add Connection window is displayed.
3. Change the Data Source selection to the Virtual View Manager server.
   
   If the Virtual View Manager server is not displayed, then either the driver was not installed, or Visual Studio must be restarted to recognize the driver.
4. Click OK to return to the Add Connection window and continue configuring the Virtual View Manager Data Provider.
   
   • Virtual View Manager Host—Virtual View Manager Server host name or IP octet. This value would be localhost if the Visual Studio development will be performed locally on the same computer as the Virtual View Manager server.
   
   • Port—Use the number of either the JDBC/ODBC open port (default: 9401) or the SSL protected port (default: 9403). For use of SSL, Server certificates must be installed separately. See the "SSL Management" topic in the "Post-Installation" chapter of the *IBM Cognos Virtual View Manager Administration Guide* for more information.
   
   • User Name / Password / Domain—These fields refer to the Virtual View Manager administrator's user name, password, and the domain to which the administrator belongs, normally that domain is "cognos" for installations with locally defined users.
   
   The user name, password, and domain profile is used to mediate communication with the hosted data service. This profile does not authorize, nor give rights and privileges to ODBC clients for use of data or services other than connection and view of the published Virtual View Manager services catalog.
   
   • Datasource—Data source refers to the Virtual View Manager database name published in the Virtual View Manager Data Services node.
5. Test the connection and click OK when finished with the configuration.

**Advanced connection properties**

Advanced Connection properties may be configured to change default connection timeout, fetch limits, reconnection, and failover settings.
• ConnectTimeout—number of seconds the client will wait for a connection to be established or to fail. A value of 0 disables the timeout.

• FetchBytes—the maximum number of bytes fetched from Virtual View Manager at one time.

• FetchRows—the maximum number of rows fetched from Virtual View Manager at one time.

There is no relationship between FetchRows and FetchBytes. When the dbchannel gets a record, it will calculate fetched rows and fetched bytes, if fetched rows or fetched bytes is bigger than fetchRows or fetchBytes, dbchannel will stop fetch row and return all fetched records.

• RequestTimeout—number of seconds the client will wait for the Virtual View Manager server to return a request. A value of 0 disables the timeout.

• SessionTimeout—timeout for session inactivity on the server; This setting gives the Virtual View Manager server an indication of how long a connection should be maintained if the connection with the client is lost without the server being notified.

• EnableFailover—Default value is false which will result in an exception if the connection object dies and a new connection to the same server will be created.

• Connection—properties in this section are values that were set on the original Add Connection window.

• Debug TraceLevel—enables or disables debug level logging.

• Connection Lifetime—sets the duration that an inactive connection (a connection that was returned to the pool) will persist 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.

• Maximum Pool Size—sets the maximum number of connections that will be opened in the same pool at the same time. If the maximum is reached and no usable connection is available, then subsequent requests are queued until a connection is available.

• Minimum Pool Size—sets the minimum number of connections that will be maintained even if inactive to avoid the time cost of recreating new connections for a new request.

• Pooling—when true inactive connections are saved and reused as necessary.

• Password—is the same as was entered on the Add Connection window.

• ValidateRemoteCert—is not supported by ADO.NET.

After properly defining the Connection profile settings the newly created connection is displayed in the Visual Studio Server Explorer. Now you can work with the Virtual View Manager server using the standard Server Explorer interface.

After a connection is successfully established, all the connection settings are saved. When you next open Visual Studio, the connection to the Virtual View Manager server will display within Server Explorer Server Host. Virtual View Manager services are ready for ADO.NET client usage.

Modifying or deleting a connection

To modify and delete a connection, use the Visual Studio Server Explorer context menu for the corresponding node. You can modify any of the settings just by overwriting the existing values with new ones. Note that a connection should be
modified or deleted only if no active editor for its objects is opened. Otherwise your data could be lost.

**Connecting to IBM Cognos Virtual View Manager server using SOAP**

Web service clients can access IBM Cognos Virtual View Manager defined web services published via SOAP over HTTP, SOAP over WSIF JMS, or SOAP over TIBCO JMS.

View the WSDL of a Virtual View Manager Web Service published by SOAP over HTTP.

**Procedure**

Do one of the following:
- Right click select the Virtual View Manager Data Service, and click View WSDL.
  A new browser window is opened to display the currently published WSDL.
- In a Web browser, enter the URL to connect to the Web service:
  http://HostName:9400/services/FolderName/DataServiceName.wsdl
  where
  HostName is the name of the machine where Virtual View Manager server is running. If you are on the same computer server as the installation of Virtual View Manager server then the computer name would be localhost.
  FolderName is any optional folder of set of folders that contains the Virtual View Manager Data Service.
  DataServiceName is the name for the WSDL-transformed data service.
  This URL (and any accessible WSDL URL) may be used as a data source to demonstrate the availability of the data or to demonstrate Virtual View Manager introspection into WSDL data sources.
  The WSDL of the Data Service is displayed in the browser.

The Web service is now ready for importing into your client application.

**Consumption of SOAP over JMS services**

Client side applications seeking to consume services published over JMS require the following:
- WSDL and the binding specified by the WSDL
- JMS Connector name
- Message pipeline information like signature and encryption certificates, incorporating the public certificate of the IBM Cognos Virtual View Manager Server.

**Note:** If Virtual View Manager resources are published over an ESB JMS broker, the client consuming the service must set a JMS application property named Authorization and set to

Basic <base64> user:password[@domain]

where the [@domain] value does not have to be specified if it is cognos.

Refer to the TIBCO documentation for details on TIBCO JMS binding protocols.
Apache WSIF bindings are described here:

http://ws.apache.org/wsif/providers/wsd1_extensions/jms_extension.html
Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.
Trademarks

IBM, the IBM logo, ibm.com, and Cognos are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:
• Netezza is a registered trademark or trademark of Netezza Corporation, an IBM Company.
• Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
• Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
• UNIX is a registered trademark of The Open Group in the United States and other countries.
• Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
## Index

### A
- Access data source, adding 44
- Access Tools right 156
- Actor 170
- Add button
  - procedure editor 93
- Add Username Token 170
- Add/Remove Resources window 57
- adding
  - and removing data source resources 57
  - adding parameters, overview 113
- Additional classpath, field, data source window 56
- Admin API, Read All Config right 156
- administration
  - APIs 11
- Administrator template 155
- Advanced
  - data source properties 30
- AGGREGATION
  - node query execution plan 182
- API
  - for administration 11
- APPEND 170
- Apply changes recursively 161
- Apply changes recursively to child resources and folders 161
- array, input parameter SQL script 98
- audience of document ix
- Authentication Domain 35

### B
- back-up
  - full server 24
- Base URL for Schema Locations 129, 133
- Basic XML to Tabular Mapping 100
- binding profile type 149
- binding resources 21
- bindings
  - Web services 45
- built-in procedures 5
- optimizer 189

### C
- cache
  - refreshing 195
- caching
  - enabling 195
  - setting up 195
  - views 193
- Caching panel
  - procedure editor 95
- Cancel, button 95
- CAR file 17
- CAR file restoration
  - Modify All Config right 157
- cardinality 116
- caseSensitive 224
- Certificate Alias 173
- certificates 168
- Change Password, menu item 7
- Change Resource Owner 7
- Character Set 53
- child 2
- child objects 162
- Choose Resources, menu item 7
- CLASSPATHs, multiple 56
- clearing caches, Modify All Status right 157
- client connectivity, information 223
- Close Active Editor, menu item 223
- Close All, menu item 7
- cognos/all group 156
- cognos/anonymous user 156
- column projections
  - designing from the Column panel 88
- Configuration, menu item 8
- Connection Pool Maximum Size 30
- Connection Pool Minimum Size 30
- Connection Pool Timeout 30
- connection URL 61
- Connection URL Pattern 30, 35
- Connection URL String 30
- connectivity 223
- connectTimeout 224
- container 2
- container objects 162
- container/leaf/parent/child vs. resource 2
- copy and paste 157
- Copy, toolbar item 8
- cost-based optimizer
  - built-in procedures 189
- Create Cache 195
- Create Element 172
- Create Link And Target button 102
- Create Link button 102
- Create Stats 186
- creating
  - resources 11
  - trigger 141
  - creating a view 81
  - creating from a schema 133
  - creating, information 55
- CROSS JOIN
  - node query execution plan 182
- Cursor
  - Select Mode 39
  - custom
    - Java procedures 55
- Custom Functions, menu item 8
- Custom, pipeline step 172
- Cut, toolbar item 8

### D
- Data Map panel 93
  - using 102
- data retrieval
  - mechanisms 223
  - data source
  - add/remove resources 57
  - adding 35
data source (continued)  
connection validation query  30
supported types  29
what is  28
wizard  28
data source testing, Modify All Status right  157
data sources  
adding  28
auto discovery  27
drivers  61
introspecting  28
on network  27
types, supported  27
data types  
supported  98
supported in definition sets  127
database connectivity  
JDBC  223
ODBC  225
Databases  
node  4
DB2 data source  
adding  30
DDL, generating and executing  195
default rights and privileges  154, 155
definition  
use  127
definition set  127, 133, 135
creating  130
SQL-type  131
XML-type  48
Delete Element  173
Delete, toolbar item  8
delimited file  
adding  53
Delimiter  53
dependent resources  21
description of product ix
design by example  118
Design Mode  117
procedure editor  93
design-time privileges  162
Desktop  3
DIGEST  170
digital certificates  168
Direct  
Select Mode  39
DISTINCT  
node query execution plan  182
Do not gather statistics  186, 187
drivers  
data sources  61
dynamic/all group  156

E  
editing  
resources  15
stored procedures  117
editor  
procedure  91
editors  
and input dialogs  9
Element Name  173
Enable Data Source menu item  7
enable trigger  140
Enable Trigger, menu item  7
crypt  224
Encrypt Element  173
Encryption Algorithm  173
ERP data sources  61
event log display  8
examples  
in Databases, resource tree  4
in Shared, resource tree  5
Excel data source, adding  45
execute procedure  
trigger action  140
Execute SELECTs Independently  30
executing  
procedures  121
Execution Plan screen  182
Export  
menu item  7
export file  17
Export window  19
exporting  
definition  17
multiple resources  19
through Studio  18
External Schema Location  129
syntax  133

F  
FETCH  
node query execution plan  182
fetchBytes  224
fetchRows  224
file  53
data source, adding  52
JAR, path  56
XML, adding  55
File Mode  170
File Name Filters  53
File Path  170
Filter  
Privileges window  161
FILTER  
node query execution plan  182
filters, filename  53
Find/Replace, toolbar item  8
Footer  170
Foreign Key  192
FULL OUTER JOIN  
node query execution plan  182
full server backup  
Read All Users right  158
Full Server Backup, menu item  8
Full Server Backup, Read All Resources right  157
FUNCTION  
node query execution plan  182

G  
Gather all statistics  186, 187
Gather column boundary statistics  186, 187
Gather table boundary statistics  186, 187
gathering statistics  
trigger action  140
global inputs, to XQuery  107
GoTo Line, toolbar item  8
Grant Privilege 166
Grid panel 92
GROUP BY
  node query execution plan 182
group privileges
  managing 161
group rights
  LDAP and iPlanet 155
Group Rights Template 155
groups
  pre-created anonymous user 156
groups and users
  modify rights 158

H
Has Header Row 53
HASH, join algorithm 178
Header 170
Help Topics
toolbar item 8
home folder 6
host name
  node, resource tree 5

I
ignoreTrailingSpace 224
import file 17
Import, menu item 7
importing
  definition 17
  resources 21, 22
  rules for 22
IN
  node query execution plan 182
Include Resource Information
  for exporting 19
indexes
  defining and viewing 191
  primary 191
  unique 191
Indexes panel 191
Info panel
  procedure editor 95
  trigger 141
Informix data source
  adding 38
  input dialogs
    and editors 9
Inputs panel 94
installing
  SAP JCo AIX 66
  SAP JCo HP-UX 67
  SAP JCo Linux 66
  SAP JCo Solaris 66
introspection
  definition 28
  scenario 28
iPlanet user rights 155

J
JAR
  path 56
Java Key Store and PKCS 168
Java procedures
  adding 55
  creating, information 55
JDBC 223
  connection multiple credentials 225
  connectivity 223
  unsupported methods 225
JDBC Connection Properties 224
JDBC connection URL 223
JDBC/ODBC
  information 223
JMS Queues 149
JOIN
  order enforcing 181
  properties specifying 177
  node query execution plan 182

K
keystore 168

L
languages in procedures 91
Launch Administrator 8
LDAP
  data source 45
  LDAP user rights 155
  leaf 2
  leaf objects 162
LEFT OUTER JOIN
  specifying 177
lib
  node, resource tree 5
License Management, menu item 8
Local File System 53
Lock creation time 13
Lock owner 13
Lock owner domain 13
Lock type - Explicit 13
Log Message To File 170

M
Manage Domains, Groups, and User membership and
  rights 8
  maximum number in queue 141
menu bar
  Modeler 7
  menu options, right-click 11
message level security 169
Message Pipeline 170
Microsoft Access data source, adding 44
Microsoft Excel data source, adding 45
  model
    designing from SQL 87
Model panel
  procedure editor 92
  transformation editor 104
Modify 157
Modify All Config 157
Modify All Config right 157
Modify All Resources right 157
Modify All Status right 157
modify all users 14
Modify All Users right 158
moving resources 16
multi-pass-through credentials 225
multiple CLASSPATHs 56
Must Understand 170
My Home
node, resource tree 4
MySQL data source
adding 40

N
Namespace 173
Netezza data source
adding 40
New, menu item 7
No Name Space Schema Location 55
nobody user 156

O
ODBC
connectivity 223, 225
ODBC/JDBC
information 223
Open, menu item 7, 8
opening resources 12
operation-level security settings 169
operations folder 148
Oracle 40
Oracle data source
adding 40
Oracle OCI Driver data source
adding 41, 42
ORDER BY
node query execution plan 182
original resources
published resources 147
Outputs panel 94
Outputs, panel 108
OVERWRITE 170

P
packaged queries 108
creating 108
data source associated 108
specifying input parameters constraints 109
panel 128
panels
procedure editor 92
Parameter Mapping panel 47
Parameter Values 173
parameterized query 113, 114
creating 113
parameters 115
parameters
adding to a parameterized query 113
defining 93
global xquery 107
including in the SELECT clause 114
input 115
names restrictions 115
parameters input
including in SELECT 114
including in WHERE 115
Parameters panel
packaged query 108
procedure editor 93
parent 2
parent objects 162
Parent XPath 173
pass-through credentials 30, 195
adding/removing resources 57
non-pass-through mode options 35
pass-through mode options 35
row-based security 35
Pass-through credentials
pass-through mode options 30
row-based security 30
Pass-through Login 30
pass-through credentials 35
password
changing 167
Paste, toolbar item 8
path
file, network 53
LDAP 45
WSDL 46
physical stored procedures 116
pipeline 169
pipeline steps 169
port
configuration 149
port-level security settings 169
pre-created users and groups 156
Prefix 173
Pretty Print 170
primary index 191
primary key 191
Privilege
Grant privilege and resource ownership 166
privileges
display 165
enforcement 162
propagation 162
resource 162
Privileges 154, 159
Default privileges for new resources 159
Design-time privileges 160
Granularity 165
Read and Write 160
Run-time privileges 160
Select, Execute, Insert, Update, and Delete 160
Setting privileges 161
Privileges, menu item 7
PROCEDURE
node query execution plan 182
nodes 185
procedure editor 91
panels 92
procedure type
panels displayed 92
procedures
adding to the Model panel 92
built-in, system 5
definition 91
executing 121
languages 91
transaction options setting 120
types 91
Process Security Header 174
propagation of privileges 162
publish
 how 149
 where 148
 Publish
 menu item 7
 toolbar item 8
 published resources
 access privileges 151
 usage 151
 publishing
 multiple resources 149
 publishing data services 149
 purpose of document ix

Q
 query execution
 plan 182
 statistics 185
 update plan 185

R
 re-introspect data source
 trigger action 140
 re-introspection
 definition 28
 process 57
 why 29
 Read All Config 156
 Read All Config right 156
 Read All Resources right 157
 Read All Status right 157
 Read All Users right 158
 Read Committed 30
 Read Uncommitted 30
 Rebind panel
 procedure editor 95
 rebindables, marking 21
 rebinding
 procedures 120
 Refresh All, menu item 7
 Refresh All, toolbar item 8
 Refresh, menu item 7
 refreshing
 cached views 195
 Reopen, menu item 7
 Repeatable Read 30
 Replace, part of Find, toolbar item 8
 requestTimeout 224
 Reset of the System Namespace, Read All Users right 158
 Resource locking
 Change History 14
 Resource Names 2
 resource privileges 162
 resource tree 2
 system-created nodes 2
 resource vs. container/leaf/parent/child 2
 resources
 binding 21
 copying 17
 creating 11
 creating catalog 12
 creating folder 12
 creating port 12
 creating schema 12
 resources (continued)
 creating Virtual View Manager databases, Web services 12
 creating Web service 12
 deleting 17
 editing 15
 exporting multiple 19
 importing 21, 22
 moving 16
 opening 12
 published, privileges 165
 publishing 147
 renaming 17
 saving 15
 searching for 24
 where to create 12
 Result mapping 70
 import and export parameters 70
 table parameters 70
 Result panel 25
 procedure editor 95
 RETURN, VALUE, to be deleted 117, 118
 RIGHT OUTER JOIN
 node query execution plan 182
 specifying 177
 rights
 Access Tools 156
 iPlanet 155
 LDAP 155
 Modify All Config 157
 Modify All Resources 157
 Modify All Status 157
 Modify All Users 158
 Read All Config 156
 Read All Resources 157
 Read All Status 157
 Read All Users 158
 Rights 154, 155
 rights and privileges
 default 154, 155
 Root Path 53
 runtime privileges 162

S
 Salesforce.com data source
 adding 61
 capabilities 62
 SAP ABAP queries and infosets
data type mapping 70
 SQL capabilities 70
 supported operators 71
 SAP data source
 adding 63
 resources 67
 SAP functions 68
 capabilities 69
 parameter mapping 68
 SAP Java Connector Library
 installing
 SAP JCo on UNIX or Linux 66
 SAP JCo on Windows 65
 installing UNIX or Linux 66
 installing Windows 65
 SAP Tables
 capabilities 68
 SQL support 67
Save
  menu item 7
toolbar item 8
Save All, menu item 7
Save As, menu item 7
Save Password 30
  pass-through credentials 35
saving resources 15
schema and contents, viewing 25
Schema Location, syntax 55
searching
  resources 24
security 167
  certificates 168
  column-based 25
  message level 169
  service port 168
  transport layer 168
web services 167
security features 153
security settings 169
SELECT
  node query execution plan 182
Select Mode 39
Semijoin 180
  manual specification 180
  Non-Equi 181
  partitioned semijoin 180
SEMIOJOIN, join algorithm 180
send e-mail
trigger action 140
Serializable 30
server
  back-up from studio 24
Server Logs 8
  services, resource tree 5
sessionTimeout 224
Set Environment From Node 175
Set Node From Environment 175
Shared, node 5
shared, resource tree 6
Show All Databases 39, 43
Show All Properties 128
Show Contents 25
Show Contents, menu item 7
Show Dependencies, menu item 7
Siebel data source
  adding 71
  installing Siebel Data Bean JARs 72
  object mapping 73
SQL capabilities 73
Sign Element 176
SOAP
  connectivity 223
SORTMERGE, join algorithm 179
Source Alias column, Model panel 104
Source Alias, column 104
Source column, Model panel 104
source scope, output XML document 104
Source Settings column 104
Source Settings column, Model panel 104
Source, column 104
SQL
  execution plan 182
SQL panel 93
  packaged query 108
SQL Script
  creating 97
  Parameters panel 98
SQL Script panel 92
SQL Server data source
  adding 39
standards supported 167
Storage 187
stored procedures
  cursor 118
  defining cursor for projection 117
  direction unknown 117
  executing 124
streaming transformation 101
Switch User, menu item 7
Sybase data source
  adding 42
system
  in Databases, resource tree 4
  in Web Services, resource tree 4
  services 4
tables 4
system event
  trigger 140, 143
system user 156
T
table
  schema and contents, viewing 25
  viewing panel 25
tabular data into XML data 104
Target 104
Target column, Model panel 104
target schema 101
Target Value 104
Target Value column, Model panel 104
Target, column 104
Teradata 43
Teradata JDBC driver 43
terminating sessions, Modify All Status right 157
Test Service, menu item 7
testing
  Web service 152
TEXT 170
Text Qualifier 53
timer event trigger 142
toolbar
  procedure editor 96
  view editor 129
traceLevel 224
Transaction Isolation levels 30, 35
transaction options
  setting for procedures 120
transaction options setting, for procedures 95
transformation
  creating non-XQuery type 102
  creating, XQuery type 104
designing target columns Outputs panel 103
  executing 121
tabular to XML and vice versa 100
types 100
transport layer security 168
trigger
  creating 141
editor 139
  system event 140
trigger (continued)
  timer event  140
  user-defined event  140, 143
trigger action types
  action types for a trigger  140
trigger condition types
  condition types for a trigger  140
triggers
  uses and benefits  139
types of procedures  91

U
UNION
  node query execution plan  182
unique index  191
unlock resources  14
user rights
  LDAP and iPlanet  155
User Rights Template  155
users
  node, resource tree  6
  pre-created  156

V
Variable Name  175
view
  creating  81
  dependencies  90
  designing  81
  designing in the Grid panel  84
  designing in the Model panel  81
  designing in the SQL panel  87
details  88
executing  89
rebinding  89
View Editor  75
  Caching panel  78
  Columns panel  78
  Foreign Keys panel  78
  Grid panel  76
  Indexes panel  78
  Info panel  78
  Model panel  75
  panels  75
  Rebind panel  79
  Result panel  79
  SQL panel  77
toolbar  79
View WSDL, menu item  7
viewing
  table schema and contents  25
views
  about  75
  Virtual View Manager Data Services
    node, resource tree  3

W
Web service
  testing  152
web services
  security standards supported  167
Web Services
  location for publishing  148
  node, resource tree  4
web services, security  167
ws
  security  167
  supported security standards  167
WSDL
  encoding scheme  45
  mapping parameters to message parts  47
  message style  45
  path  46
  transport  45
  viewing  152
WSDL data source
  adding  46
details  45
  using  47

X
XML data source, using  55
XML definition set  128
XML schema  136
XML Schema  128
XML target schema  101
XML-type  133, 135
XML-Type definition set  135, 136
XML/HTTP data source
  adding  48
XPath  175
XQuery panel  94
XQuery procedure  101
xquery transformation
  specify settings  107
XQuery transformation  101
XQuery, panel  108
XSD
  location  133
XSLT Debug Input panel  94
XSLT Debug Output panel  95
XSLT panel  94
XSLT Transformation  101